PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-293709
(43)Date of publication of application : 20.10.2000

(51)Int.Cl. G06T 17/00
G06T 11/00
G06T 15/00

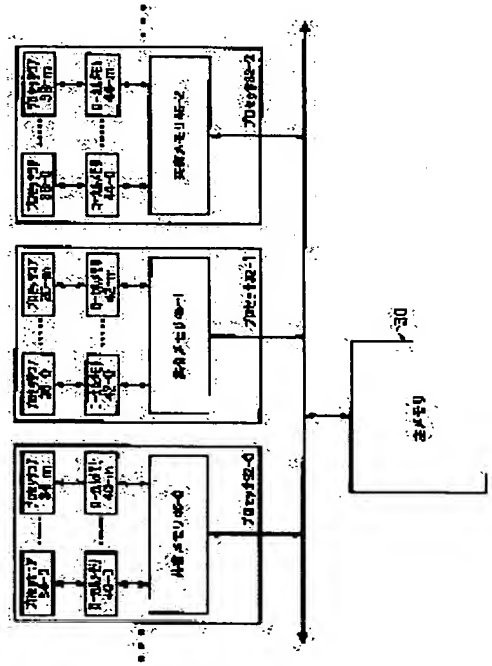
(21)Application number : 2000-022197
(22)Date of filing : 31.01.2000

(71)Applicant : TOSHIBA CORP
(72)Inventor : KUNIMATSU ATSUSHI
UENO KIYOJI
YASUKAWA HIDEKI
WATANABE YUKIO
KAMEI TAKAYUKI
AMATSUBO TAKANAO

(30)Priority
Priority number : 11026563 Priority date : 03.02.1999 Priority country : JP

(54) DEVICE, SYSTEM AND METHOD FOR PROCESSING PICTURE

(57)Abstract:
PROBLEM TO BE SOLVED: To realize a high degree of picture processing with an inexpensive system by improving the capacity of a computer which makes the picture processing.
SOLUTION: A picture processor is composed of a main memory 30 which stores information on three-dimensional objects, a plurality of processor cores 34 (34-0 to 34-m), 36 (36-0 to 36-m), and 38 (38-0 to 38-m), and local memories 40 (40-0 to 40-m), 42 (42-0 to 42-m), and 44 (44-0 to 44-m) and shared memories 46 (46-0 to 46-2) which have a plural tree and are connected between the main memory 30 and processor cores 34, 36, and 38 and successively store a part of the lower-level information at higher levels. The picture processor performs picture processing in parallel by means of the processor cores 34, 36, and 38.



LEGAL STATUS

[Date of request for examination] 28.01.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of

rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号
特開2000－293709
（P2000－293709A）
(43)公開日　平成12年10月20日(2000.10.20)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｇ０６Ｔ 17/00		Ｇ０６Ｆ 15/62		３５０Ａ
11/00		15/72		３５０
15/00				４５０Ａ

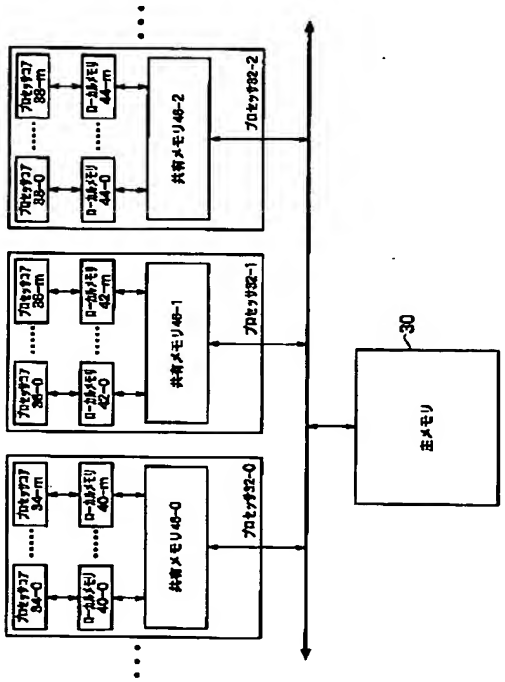
審査請求　未請求　請求項の数16　ＯＬ　（全 31 頁）

(21)出願番号　特願2000－22197(P2000－22197)
(22)出願日　平成12年１月31日(2000.1.31)
(31)優先権主張番号　特願平11－26563
(32)優先日　平成11年２月３日(1999.2.3)
(33)優先権主張国　日本（ＪＰ）

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　国松　敦
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝マイクロエレクトロニクスセンター内
(72)発明者　上野　喜代治
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝マイクロエレクトロニクスセンター内
(74)代理人　100083806
弁理士　三好　秀和　（外７名）
最終頁に続く

(54)【発明の名称】　画像処理装置、画像処理システムおよび画像処理方法

(57)【要約】
【課題】　画像処理する計算機の処理能力を向上させ、高度な画像処理を安価なシステムで実現可能な画像処理装置、画像処理システムおよび画像処理方法を提供する。
【解決手段】　３次元物体の情報を格納する主メモリ３０、主メモリ３０から読み出された３次元物体情報に基づいて画像処理する、複数のプロセッサコア３４，３６，３８、複数の階層を有し、主メモリ３０とプロセッサコア３４，３６，３８との間に接続され、下位レベルの情報の一部を上位レベルに順次格納する、ローカルメモリ４０，４２，４４、共有メモリ４６、から成る画像処理装置である。この画像処理装置は、複数のプロセッサコア３４，３６，３８による画像処理を並列処理する。



1

【特許請求の範囲】

【請求項１】　３次元物体の情報を格納する主記憶部と、

前記主記憶部から読み出された３次元物体情報に基づいて画像処理する、複数の演算部と、

複数の階層を有し、前記主記憶部と前記演算部との間に接続され、下位レベルの情報の一部を上位レベルに順次格納する階層記憶部とを具備し、

前記複数の演算部による画像処理は、並列処理される、ことを特徴とする画像処理装置。

【請求項２】　前記階層記憶部の最上位レベルを構成する記憶部は、対応する演算部に従属し、かつ高速アクセス可能な、複数の従属記憶部であり、

前記主記憶部は、前記演算部に共有される、ことを特徴とする請求項１に記載の画像処理装置。

【請求項３】　前記３次元物体の情報は、前記３次元物体を包含する空間を、複数の階層で構成された、複数の小空間に分割された状態で、前記主記憶部に格納される、ことを特徴とする請求項２に記載の画像処理装置。

【請求項４】　３次元物体の情報を格納する主記憶部と、前記主記憶部から読み出された３次元物体情報に基づいて画像処理する、複数の演算部と、複数の階層を有し、前記主記憶部と前記演算部との間に接続され、下位レベルの情報の一部を上位レベルに順次格納する階層記憶部と、を有する、複数の画像処理装置と、

該複数の画像処理装置を相互に接続する通信媒体とを具備し、

前記複数の演算部による画像処理は、並列処理される、ことを特徴とする画像処理システム。

【請求項５】　前記階層記憶部の最上位レベルを構成する記憶部は、対応する演算部に従属し、かつ高速アクセス可能な、複数の従属記憶部であり、

前記主記憶部は、前記演算部に共有される、ことを特徴とする請求項４に記載の画像処理システム。

【請求項６】　前記３次元物体の情報は、前記３次元物体を包含する空間を、複数の階層で構成された、複数の小空間に分割された状態で、前記主記憶部に格納される、ことを特徴とする請求項５に記載の画像処理システム。

【請求項７】　３次元物体の画像処理を分割する工程と、

該分割された複数の処理を、複数の演算部に配分する工程と、

該複数の演算部で、配分された処理を、並列に行なう工程とを少なくとも含み、

前記３次元物体の情報は、前記３次元物体を包含する空間を、複数の階層で構成された、複数の小空間に分割された状態で、主記憶部に格納される、ことを特徴とする画像処理方法。

【請求項８】　前記３次元物体は、

2

３次元座標軸および時間軸で定義された４次元空間に存在すると共に、

複数の点で定義された第１の平面図形と、複数の点で定義された、複数の第２の平面図形の組み合わせで定義された第１の立体図形と、少なくとも１つの関数で定義された第２の立体図形、のうちの少なくとも１つである、ことを特徴とする請求項３に記載の画像処理装置。

【請求項９】　前記３次元物体は、

３次元座標軸および時間軸で定義された４次元空間に存在すると共に、

複数の点で定義された第１の平面図形と、複数の点で定義された、複数の第２の平面図形の組み合わせで定義された第１の立体図形と、少なくとも１つの関数で定義された第２の立体図形、のうちの少なくとも１つである、ことを特徴とする請求項６に記載の画像処理システム。

【請求項１０】　前記従属記憶部のデータ領域は、３次元空間の階層構造を格納する領域と、３次元物体情報を格納する領域と、に論理的または物理的に分割される、ことを特徴とする請求項３に記載の画像処理装置。

【請求項１１】　前記従属記憶部のデータ領域は、３次元空間の階層構造を格納する領域と、３次元物体情報を格納する領域と、に論理的または物理的に分割される、ことを特徴とする請求項６に記載の画像処理システム。

【請求項１２】　前記複数の演算部間を接続する通信媒体は、複数の演算部が、同一または近傍のアクセス領域にアクセス要求する場合には、それらアクセス要求を結合する、待ち行列手段を備える、ことを特徴とする請求項１に記載の画像処理装置。

【請求項１３】　前記複数の演算部間を接続する通信媒体は、複数の演算部が、同一または近傍のアクセス領域にアクセス要求する場合には、それらアクセス要求を結合する、待ち行列手段を備える、ことを特徴とする請求項４に記載の画像処理システム。

【請求項１４】　視点から画面上の１つの画素を通過する半直線と、３次元物体と、を交差判定し、該判定の結果から前記画素の画像処理をする画像処理方法において、

３次元空間内の複数の３次元物体の中から１つ選択し、該選択された３次元物体の情報を取得する第１の工程と、

該取得された情報に基づいて、前記画面上のすべての画素について交差判定する第２の工程と、

前記３次元空間内のすべての３次元物体に対して、前記第１および第２の工程を実行する第３の工程とを少なくとも含むことを特徴とする画像処理方法。

【請求項１５】　３次元物体の連続画像を生成する画像処理方法において、連続画像を表示する画面の１コマを分割する工程と、

該分割された複数の領域を、複数の演算部に割り当てる工程と、

該複数の演算部で、割り当てられた領域を、並列処理する工程と、
該処理の結果に応じて、前記複数の演算部それぞれの処理時間が等しくなるように、次の１コマを分割する工程とを少なくとも含むことを特徴とする画像処理方法。
【請求項１６】　３次元物体の画像を生成する画像処理方法において、
画像を表示する画面を分割する工程と、
該分割された複数の領域を、複数の演算部に割り当てる工程と、
該複数の演算部で、割り当てられた領域を、並列処理する工程とを少なくとも含み、
前記並列処理する工程は、割り当てられた領域の処理が終了した演算部は、他の演算部に割り当てられた未終了の領域を処理する工程を含む、ことを特徴とする画像処理方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、３次元コンピュータグラフィックスを生成する画像処理装置に係り、特に、複数の計算機を用いて並列に画像処理する画像処理装置、この画像処理装置を用いた画像処理システム、およびその画像処理方法に関する。
【０００２】
【従来の技術】コンピュータで文字や静止画、動画（アニメーション）を加工する画像処理は、その画像データの大きさから、非常に負荷の大きい処理となる。そのため、レンダリング処理等の画像処理は、スーパーコンピュータやハイエンドワークステーション等の高価な設備を必要とする。
【０００３】一方、３次元物体をポリゴンに分割し、ポリゴンを処理単位として描画する画像処理は、安価なシステムで実現可能である。しかしながら、高精細で質感のある映像を描画しようとすれば光の反射や屈折、煙などの粒子の動き等を再現しなければならない。すなわち、物体があたかも３次元空間に存在するように描画するためには、物体の配置（遠くにある物の一部は、近くにある物によって隠れる）や、光線の加減（光源に面した部分は明るく、光源と反対の面は暗い）、物体の材質（鏡面なら光源からの光を反射し、透明の物体なら光線を透過させる）などを考慮し、それらによって物体がどのように見えるかを計算しなければならない。その結果、高価なシステムや膨大な処理時間が結局必要となる。
【０００４】
【発明が解決しようとする課題】近年、今まで高価なスーパーコンピュータやワークステーションによらなければ得られなかった高精細かつ質感のある３次元グラフィックスを、安価なシステムで実現することが強く望まれるようになって来ている。
【０００５】本発明は、上述の如き事情に鑑みて成されたものであり、その目的は、レンダリング処理の高速化および高効率化を図り、低コストのシステムを実現できる画像処理装置を提供することである。
【０００６】本発明の他の目的は、画像処理の高速化、高効率化を実現する、複数の画像処理装置、を含む画像処理システムであって、低コストのシステムを提供することである。
【０００７】本発明のさらに他の目的は、レンダリング処理の高速化および高効率化を図り、低コストのシステムを実現できる画像処理方法を提供することである。
【０００８】
【課題を解決するための手段】上記課題を解決するため、本発明は、３次元物体の情報を格納する主記憶部と、その主記憶部から読み出された３次元物体情報に基づいて画像処理する、複数の演算部と、複数の階層を有し、主記憶部と演算部との間に接続され、下位レベルの情報の一部を上位レベルに順次格納する階層記憶部、とを少なくとも有する画像処理装置であることを第１の特徴とする。ここで、上記の複数の演算部による画像処理は、並列処理される。
【０００９】本発明の第２の特徴は、第１の特徴で述べた画像処理装置において、３次元物体の情報は、その３次元物体を包含する空間を、複数の階層で構成された、複数の小空間に分割された状態で、主記憶部に格納され、その３次元物体は、３次元座標軸および時間軸で定義された４次元空間に存在すると共に、複数の点で定義された第１の平面図形と、複数の点で定義された、複数の第２の平面図形の組み合わせで定義された第１の立体図形と、少なくとも１つの関数で定義された第２の立体図形、のうちの少なくとも１つであることである。
【００１０】本発明の第３の特徴は、第１の特徴で述べた画像処理装置において、階層記憶部の最上位レベルを構成する記憶部は、対応する演算部に従属し、かつ高速アクセス可能な、複数の従属記憶部であり、その従属記憶部のデータ領域は、３次元空間の階層構造を格納する領域と、３次元物体情報を格納する領域と、に論理的または物理的に分割される、ことである。また、従属記憶部のデータ領域が、物理的に分割される場合には、３次元物体を格納する領域は、３次元物体情報を保持するメモリ部と、メモリ部中の３次元物体情報の配置を示すテーブル部と、対応する演算部が要求する３次元物体情報が、メモリ部中に存在しない場合、メモリ部の一部と主メモリの一部とを置き換え、要求された３次元物体情報をメモリ部に読み出す制御部とを少なくとも有するように構成すれば良い。
【００１１】本発明の第４の特徴は、第１の特徴で述べた画像処理装置において、複数の演算部間を接続する通信媒体は、複数の演算部が、同一または近傍のアクセス領域にアクセス要求する場合には、それらアクセス要求

を結合する、待ち行列手段を備えることである。

【００１２】本発明の第５の特徴は、視点から画面上の１つの画素を通過する半直線と、３次元物体と、を交差判定し、その判定の結果から画素の画像処理をする画像処理方法において、３次元空間内の複数の３次元物体の中から１つ選択し、その選択された３次元物体の情報を取得する第１の工程と、その取得された情報に基づいて、画面上のすべての画素について交差判定する第２の工程と、３次元空間内のすべての３次元物体に対して、第１および第２の工程を実行する第３の工程とを少なくとも含むことである。

【００１３】本発明の第６の特徴は、３次元物体の連続画像を生成する画像処理方法において、連続画像を表示する画面の１コマを分割する工程と、その分割された複数の領域を、複数の演算部に割り当てる工程と、その複数の演算部で、割り当てられた領域を、並列処理する工程と、その並列処理の結果に応じて、複数の演算部それぞれの処理時間が等しくなるように、次の１コマを分割する工程とを少なくとも含むことである。

【００１４】本発明の第７の特徴は、３次元物体の画像を生成する画像処理方法において、画像を表示する画面を分割する工程と、その分割された複数の領域を、複数の演算部に割り当てる工程と、その複数の演算部で、割り当てられた領域を、並列処理する工程とを含むことである。ここで、その並列処理する工程は、割り当てられた領域の処理が終了した演算部は、他の演算部に割り当てられた未終了の領域を処理する工程を含む。

【００１５】

【発明の実施の形態】まず最初に、本発明に係るレンダリングおよびそのアルゴリズムであるレイトレーシングについて説明し、次に、本発明に係る画像処理装置、画像処理システム、および画像処理方法について７つの実施の形態を用いて説明する。

【００１６】（レンダリングおよびレイトレーシング）本発明に係る画像処理は、３次元グラフィック処理の１つであるレンダリング（rendering）を利用する。そして、そのレンダリングの手法として、本発明に係る画像処理は、処理の並列性が高く、かつ比較的小さいデータで処理が可能なレイトレーシング（ray tracing）というアルゴリズムを使用する。レイトレーシングでは、立方体や球等の物体は数式でモデル化されるため、取り扱われるデータ量は、各オブジェクトをポリゴンで分割する、従来の場合よりも小さくなる。また、視点からの光線を各画素に飛ばし、交差する物体の彩度や明度を計算するので、各画素の計算を独立して行うことが可能となる。

【００１７】図１は、本発明に係る画像処理で使用されるレイトレーシングを説明する概念図である。図１に示すように、光源１０から出た光は、さまざまなオブジェクト１２，１４，１６に当たり、吸収、反射、屈折されて観測者の目（視点）１８に届く。レイトレーシングでは、各オブジェクト１２，１４，１６の反射や透過などを視点１８から所定の画素２０を通る光線（レイ）２２をさかのぼって追跡する。そして、最初に交差したオブジェクト１４をその画素２０に描くべきものとし、最終的なイメージを計算する。レイトレーシングでは、各オブジェクト１２，１４，１６の輝度や、透明度、反射等が忠実に再現され、非常にリアルな画像を描写できる。しかしながら、レイトレーシングは、全ての画素についての計算が必要である、オブジェクト数が増加すると計算量が急速に増大する、等の理由から、強力な処理能力が要求されるという問題点を含んでいる。

【００１８】そこで、このレイトレーシングと呼ばれるアルゴリズムを、ハードウェアで構成することを考えてみる。図１に示したスクリーン２４は、無数の画素２０が集まって構成され、さらに、各画素２０は、並列に処理可能である。しかしながら、各画素２０を並列処理すれば、異なる画素２０の計算に同一のオブジェクトデータが同時に必要となる場合が生じ、その結果、メモリ・アクセスの競合を招くおそれがある。本発明は、幾つかの画素２０の計算には同じオブジェクトデータが使用されるというデータの局所性（locality）、すなわちメモリ・アクセスの局所性、を利用することにより、各画素２０の計算およびメモリ構造の分割・階層化を図るものである。

【００１９】本発明においては、まず、図２に示すように、図１のスクリーン２４は、複数のサブ・スクリーン２６に均等に分割される。さらに各サブ・スクリーン２６は、複数のサブ・サブ・スクリーン２８に均等に分割される。次に、図３に示すように、各種のオブジェクトデータが格納された主メモリ３０と、主メモリ３０を共有する複数のプロセッサ３２－０，３２－１，３２－２，…，３２－ｎと、からなるシステムが構成される。プロセッサ３２－０は、複数のプロセッサコア３４－０，３４－１，…，３４－ｍと、プロセッサコア３４－０，３４－１，…，３４－ｍそれぞれに従属するローカルメモリ４０－０，４０－１，…，４０－ｍと、ローカルメモリ４０－０，…，４０－ｍそれぞれと接続し、プロセッサコア３４－０，…，３４－ｍによって共有される共有メモリ４６－０と、を有している。プロセッサ３２－１は、複数のプロセッサコア３６－０，３６－１，…，３６－ｍと、プロセッサコア３６－０，３６－１，…，３６－ｍそれぞれに従属するローカルメモリ４２－０，４２－１，…，４２－ｍと、ローカルメモリ４２－０，…，４２－ｍそれぞれと接続し、プロセッサコア３６－０，…，３６－ｍによって共有される共有メモリ４６－１と、を有している。プロセッサ３２－２は、複数のプロセッサコア３８－０，３８－１，…，３８－ｍと、プロセッサコア３８－０，３８－１，…，３８－ｍそれぞれに従属するローカルメモリ４４－０，４４－

１，…，４４－ｍと、ローカルメモリ４４－０，…，４４－ｍそれぞれと接続し、プロセッサコア３８－０，…，３８－ｍによって共有される共有メモリ４６－２と、を有している。ローカルメモリ４０－０，…，４０－ｍ，４２－０，…，４２－ｍ，４４－０，…，４４－ｍは、主メモリ３０および共有メモリ４６－０，４６－１，４６－２より、小容量かつ高速アクセス可能である。以下、図示はしないが、プロセッサ３２－３，…，３２－ｎについても同様である。

【００２０】そして、図２の複数のサブ・スクリーン２６それぞれが、図３のプロセッサ３２－０，３２－１，…，３２－ｎそれぞれに割り当てられる。さらに、サブ・スクリーン２６を構成するサブ・サブ・スクリーン２８それぞれが、各プロセッサ３２－０，３２－１，…，３２－ｎ内のプロセッサコア３４－０，…，３４－ｍ，３６－０，…，３６－ｍ，３８－０，…，３８－ｍ，…それぞれに割り当てられる。たとえば図２のサブ・スクリーン２６ａが図３のプロセッサ３２－１に割り当てられ、サブ・スクリーン２６ａ内のサブ・サブ・スクリーン２８ａがプロセッサ３２－１内のプロセッサコア３６－０に割り当てられる。上記のように、各プロセッサ３２－０，３２－１，…，３２－ｎは共有メモリ４６－０，４６－１，…，４６－ｎを有しており、この共有メモリ４６－０，４６－１，…，４６－ｎは各プロセッサ３２－１，３２－２，…，３２－ｎ内の複数のプロセッサコア３４－０，…，３４－ｍ，３６－０，…，３６－ｍ，３８－０，…，３８－ｍ，…によって共有されている。各プロセッサ３２－０，３２－１，…，３２－ｎは、割り当てられたサブ・スクリーン２６の処理で使用されるオブジェクトデータを主メモリ３０から読み込む。そして、読み出されたデータを、それぞれの共有メモリ４６－０，４６－１，…，４６－ｎに格納する。さらに、各プロセッサコア３４－０，…，３４－ｍ，３６－０，…，３６－ｍ，３８－０，…，３８－ｍ，…は、割り当てられたサブ・サブ・スクリーン２８の処理で使用されるオブジェクトデータを共有メモリ４６－０，４６－１，…，４６－ｎから読み込む。そして読み出されたデータを、それぞれのローカルメモリ４０-０，…，４０－ｍ、４２－０，…，４２－ｍ，４４－０，…，４４－ｍ，…に格納する。

【００２１】このような構成によれば、各プロセッサ３２－０，３２－１，…，３２－ｎの、主メモリ３０に対するデータ・アクセス回数を抑制できる。それにより、非常に並列度の高い処理を実現できる。結果的には、高度な画像処理を効率的に行うことが可能となる。なお、上記の例では、各プロセッサ３２－０，３２－１，３２－２，…，３２－ｎ内に複数のプロセッサコア３４－０，…，３４－ｍ，３６－０，…，３６－ｍ，３８－０，…，３８－ｍ，…が存在する場合について説明したが、その数は任意である。たとえば１つのプロセッサコアであっても構わない。また、共有メモリ４６－０，４６－１，４６－２，…，４６－ｎは必ずしも必要はない。すなわち、各ローカルメモリ４０-０，…，４０－ｍ、４２－０，…，４２－ｍ，４４－０，…，４４－ｍ，…が主メモリ３０に直接接続されていてももちろん構わない。さらに、図１のスクリーン２４の分割方法においても、図２に示した例に限定されるわけではなく、最終的にスクリーン２４内の幾つかのまとまった画素を１つのプロセッサコア３４－０，…，３４－ｍ，３６－０，…，３６－ｍ，３８－０，…，３８－ｍ，…に割り当てるようにすれば良い。たとえば図２では、各プロセッサコア３４－０，…，３４－ｍ，３６－０，…，３６－ｍ，３８－０，…，３８－ｍ，…に４×４つの画素が割り当てられている。すなわち、各サブ・サブ・スクリーン２８は４×４つの画素から構成されている。

【００２２】次に、上述したレイトレーシングにOctree（オクツリー）を用いる場合について説明する。オクツリーは、レイトレーシングにおいて、各オブジェクトと光線との交差判定を効率的に行うための探索手法である。具体的には、処理対象となる３次元オブジェクト空間を所定の分割条件に基づいて分割することにより、分割後の空間の大半をオブジェクトが存在しない状態、あるいはその大部分を単一のオブジェクトが占めている状態、とする空間分割手法である。図４（Ａ）および（Ｂ）に、オクツリーの例を示す。図４（Ａ）が、２次元のオクツリーを、図４（Ｂ）が、３次元のオクツリーを示している。図４（Ａ）では、２次元空間がｘ方向に２分割、ｙ方向に２分割され、サブ空間ａ、ｂ、ｃ、ｄが形成されている。図４（Ｂ）では、３次元空間がｘ方向に２分割、ｙ方向に２分割、ｚ方向に２分割され、サブ空間ａ、ｂ、ｃ、ｄ、ｅ、ｆ、ｇ、ｈが形成されている。各方向の分割の数は上記のように２分割に限られず、それ以外でもよい。以下では、説明の簡単化を図るため、図４（Ａ）に示す２次元空間の場合を用いて説明する。

【００２３】図５は、９つのオブジェクト（obj0，obj１，obj２，…，obj８）が存在する２次元空間に対して、２次元のオクツリーを適用した図である。２次元空間の分割は次の条件に基づいている。

【００２４】（１）サブ空間内に３つ以上のオブジェクトが存在すること。

【００２５】かつ、

（２）分割階層が２レベル以下であること。

【００２６】本発明は、図５の階層的に分割された空間と図３のシステムのメモリ構造との対応づけを行うことにより、効率的なデータ・アクセスを実現する。図６に、図５の階層構造の空間をレイトレーシングに適用した例を示す。レイトレーシングでは、各オブジェクトを数式でモデル化する。したがって、データ量は各オブジェクトの大きさではなく、処理対象の空間内に存在する

オブジェクトの数に比例して増大することになる。ここで、上記のオクツリーを用いることで、各サブ空間を占めるオブジェクトの数をできるだけ少なく、理想的には１つにすることができる。すなわち、各サブ空間に存在するオブジェクトのデータ量は低減されるので、各サブ空間のオブジェクトデータ全体を、図３の各ローカルメモリ４０－０，…，４０－ｍ、４２－０，…，４２－ｍ、４４－０，…，４４－ｍ，…に格納することが可能となる。したがって、図３の各プロセッサコア３４－０，…，３４－ｍ、３６－０，…，３６－ｍ、３８－０，…，３８－ｍ，…は割り当てられた画素の計算に必要なオブジェクトデータを、それぞれのローカルメモリ４０－０，…，４０－ｍ、４２－０，…，４２－ｍ、４４－０，…，４４－ｍ，…にあらかじめ読み込むことができるようになる。さらに、各プロセッサコア３４－０，…，３４－ｍ、３６－０，…，３６－ｍ、３８－０，…，３８－ｍ，…が必要とするデータが、それぞれに従属するローカルメモリ４０－０，…，４０－ｍ、４２－０，…，４２－ｍ、４４－０，…，４４－ｍ，…に存在しない場合、つまりミスヒットの場合には、共有メモリ４６－０，４６－１，４６－２，…，４６－ｎから、その必要なデータを読み込めば良い。共有メモリ４６－０，４６－１，４６－２，…，４６－ｎには、上位の空間におけるオブジェクトデータが格納されているからである。本発明によれば、オブジェクトデータの取得を効率的に行い、それにより、交差判定の効率化・高速化を図ることができる。

【００２７】（第１の実施の形態）図７は、本発明の第１の実施の形態に係る画像処理システムの構成を示す図である。図７に示すように、本発明の第１の実施の形態に係る画像処理システムは、画像処理を行なう、複数のプロセッサコア４８を備え、この複数のプロセッサコア４８をネットワーク５０で互いに接続した、複数のチップ５２を有している。そして、複数のチップ５２を備え、この複数のチップ５２をネットワーク５４で互いに接続した、複数のボード５６を有している。さらに、複数のボード５６を備え、この複数のボード５６をバス（図示しない）で互いに接続した、複数のコンピュータ５８を有している。この複数のコンピュータ５８は、ネットワーク６０で互いに接続されている。各プロセッサコア４８は、それぞれに従属するローカルメモリ（図示しない）を備えている。各チップ５２は、それぞれに従属する共有メモリ６２を備え、その共有メモリ６２は各チップ５２内の、複数のプロセッサコア４８によって共有される。また、処理対象である３次元空間は、上述したオクツリーによって階層化されている。各共有メモリ６２は、階層化されたオブジェクトデータを格納する。

【００２８】本発明の第１の実施の形態に係る画像処理システムでは、たとえば画像１コマを描画する場合、その画像１コマは所定の面積ごとに分割される。そして、分割されたそれぞれのサブ領域の描画処理を各コンピュータ５８に割り当てる。各コンピュータ５８に割り当てられたサブ領域はさらに分割され、分割されたサブ・サブ領域の描画処理を各ボード５６に割り当てる。各ボード５６内では、割り当てられたサブ・サブ領域がさらに分割され、分割されたサブ・サブ・サブ領域の描画処理を各プロセッサコア４８に割り当てる。このように、本発明の第１の実施の形態に係る画像処理システムでは、画像１コマを小領域、すなわち幾つかの画素の集り、に分割し、各小領域の描画処理を各プロセッサコア４８に分配する。そして、各小領域の描画処理を各プロセッサコアが並列に行う。このため、画像１コマ全体についての描画処理を高速かつ効率良く行うことができる。上記の例では、画像１コマを複数のプロセッサコア４８で分担して描画処理する場合について記載したが、本発明は、これに限られることはない。たとえば動画の処理を行う場合には、各画像１コマの描画処理を各コンピュータ５８に割り当てることも可能である。

【００２９】本発明の第１の実施の形態に係る画像処理システムは、たとえば次のようにして画像処理を実行する。図８は、本発明の第１の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。図８に示すように、まず、処理対象である画像を、図２に例示したように、順次分割する（ステップＳ１０１）。そして、各プロセッサコア４８に、分割された領域の描画処理を分配する（ステップＳ１０２）。各プロセッサコア４８は、分配された領域の描画処理に必要なオブジェクトデータを主メモリや共有メモリから読み込み、その読み込まれたデータをそれぞれに従属するローカルメモリに格納する（ステップＳ１０３）。次に、各プロセッサコア４８は、割り当てられた領域の描画処理を行う。この描画処理は、各プロセッサコア４８により、並列に行われる（ステップＳ１０４）。全プロセッサコア４８の描画処理が終了すると（ステップＳ１０５ＹＥＳ）、すべての領域の描画結果から処理対象となる画像の生成を行う。生成された画像が、ディスプレイに表示されたり、画像データとして記録媒体に記憶されると（ステップＳ１０６）、画像処理が終了する。

【００３０】（第２の実施の形態）次に、本発明の第２の実施の形態について説明する。本発明の第２の実施の形態は、処理対象である画像が、静止画像とは異なり、時間とともに内容が変化する動画（アニメーション）である例を示すものである。アニメーションは、一連の連続画像を画面に表示することによって、生成される。このため、レンダリングは、時間が少しずつ異なる複数のフレームそれぞれに対して、実行されることになる。さらに、オクツリーの分割も各フレームごとに行なわなければならない。すなわち、アニメーションにおいては、フレームが変われば３次元オブジェクトの位置、形状、出現、消滅等にも変化が生じる。したがって、フレーム

が変わるたびに空間分割をやり直す必要がある。その結果、フレームの変化ごとに、主メモリ等からローカルメモリにサブ空間情報を転送し直すことが必要となる。

【００３１】本発明の第２の実施の形態においては、描画される３次元オブジェクトの情報に時間情報を付加したオブジェクト情報（以下、「４次元オブジェクト情報」と呼ぶ）を生成する。そして、この４次元オブジェクト情報に基づいて、時間方向を考慮して、オクツリーによる空間分割を行うものである。時間情報を考慮して空間分割を行うことで、フレーム変化ごとの空間分割をできるだけ少なくすることが可能となる。それにより、主メモリ等からローカルメモリへの情報転送回数を低減し、処理全体の高速化を図ることができる。以下、本第２の実施の形態について詳細に説明する。

【００３２】最初に、本発明の第２の実施の形態に係る４次元オブジェクト情報について説明する。４次元オブジェクト情報は、ｘ座標、ｙ座標、ｚ座標およびｔ座標で規定される４次元空間で定義される。４次元オブジェクト情報は、周知の３次元オブジェクト情報に時間情報を付加したものである。時間情報の例としてはたとえば次のようなものがある。

【００３３】（１）式

ｘ座標：ｘ＝３・ｔ＋４

ｙ座標：ｙ＝４・ｔ＋５

ｚ座標：ｚ＝５・ｔ＋６

時間の定義域：０≦ｔ≦１０

ここで、ｔは時間を示すパラメータである。以下も同様である。

【００３４】（２）式

ｘ・ｘ＋ｙ・ｙ＋ｚ・ｚ＋ｔ・ｔ＝１０・１０

時間の定義域：－１０≦ｔ≦１０

（３）数列

｛ｔ：ｘ｝＝｛０：－１，１：　２，２：　４｝

｛ｔ：ｙ｝＝｛０：　０，１：－１，２：－２｝

｛ｔ：ｚ｝＝｛０：　２，１：　２，２：　５｝

ここで、数列によって時間情報を格納した場合、未定義の時間は、線形補間やスプライン補間等の周知のルールによって、補間される。上記の例では、たとえばｔ＝０．５が未定義の時間である。ただし、補間することなく、すべての時間を数列化することも可能である。

【００３５】次に、４次元オブジェクト情報としてはたとえば次のようなものがある。

【００３６】（１）移動する三角形（以下、「ｏｂｊ０」と呼ぶ）

［頂点１］

ｘ座標：ｘ＝１

ｙ座標：ｙ＝１

ｚ座標：ｚ＝１－ｔ

［頂点２］

ｘ座標：ｘ＝２

ｙ座標：ｙ＝１

ｚ座標：ｚ＝１－ｔ

［頂点３］

ｘ座標：ｘ＝１

ｙ座標：ｙ＝２

ｚ座標：ｚ＝１－ｔ

ここで、時間の定義域は０．５≦ｔ≦２である。

【００３７】（２）だんだんと小さくなって消える球（以下、「ｏｂｊ１」と呼ぶ）

ｘ・ｘ＋ｙ・ｙ＋ｚ・ｚ＋ｔ・ｔ＝１

ここで、時間の定義域は０≦ｔ≦１である。

【００３８】（３）移動する球（以下、「ｏｂｊ２」と呼ぶ）

（ｘ－１．１）・（ｘ－１．１）＋（ｙ－１．１）・（ｙ－１．１）＋（ｚ－（（－ｔ／１００）－１．５））・（ｚ－（（－ｔ／１００）－１．５））＝０．０１・０．０１

ここで、時間の定義域は０≦ｔ≦２である。

【００３９】（４）移動する四面体（以下、「ｏｂｊ３」と呼ぶ）

［頂点１］

ｘ座標：｛ｔ：ｘ｝＝｛－２：－１，２：－１｝

ｙ座標：｛ｔ：ｙ｝＝｛－２：－１，２：－１｝

ｚ座標：｛ｔ：ｚ｝＝｛－２：－１，２：－０．５｝

［頂点２］

ｘ座標：｛ｔ：ｘ｝＝｛－２：－２，２：－２｝

ｙ座標：｛ｔ：ｙ｝＝｛－２：－１，２：－１｝

ｚ座標：｛ｔ：ｚ｝＝｛－２：－１，２：－０．５｝

［頂点３］

ｘ座標：｛ｔ：ｘ｝＝｛－２：－１，２：－１｝

ｙ座標：｛ｔ：ｙ｝＝｛－２：－２，２：－２｝

ｚ座標：｛ｔ：ｚ｝＝｛－２：－１，２：－０．５｝

［頂点４］

ｘ座標：｛ｔ：ｘ｝＝｛－２：－１，２：－１｝

ｙ座標：｛ｔ：ｙ｝＝｛－２：－１，２：－１｝

ｚ座標：｛ｔ：ｚ｝＝｛－２：－２，２：－１．５｝

ここで、時間ｔ＝－２とｔ＝２との間は、たとえば線形補間を行えばよい。

【００４０】次に、上記の４次元オブジェクト（ｏｂｊ０，ｏｂｊ１，ｏｂｊ２，ｏｂｊ３）が存在する４次元空間を空間分割する場合について説明する。図９は，本発明の第２の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。図９に示すように、最初、４次元空間の設定が実行される。４次元空間の設定は、上記の４次元オブジェクトすべてが、その４次元空間に含まれるように、行われる（ステップＳ２０１）。上記のオブジェクトがすべて含まれる４次元空間（初期値）としては、たとえば次に示すものがある。

【００４１】－２≦ｘ≦２，－２≦ｙ≦２，－２≦ｚ≦２，－２≦ｔ≦２

次に、設定された４次元空間の空間分割が実行される（ステップＳ２０２）。空間分割は、予め定められた分割条件に基づいて実行される。分割条件としてはたとえば次のようなものがある。

【００４２】（１）分割対象となる空間に、所定の数以上のオブジェクトが存在すること。

【００４３】かつ

（２）最大分割階層レベルを超えていないこと。

【００４４】ここでは、「所定の数」を“３”、「最大分割階層レベル」を“３”、とする。初期値として設定された、上記の４次元空間には上記のオブジェクト（obj0, obj1, obj2, obj3）が４つ存在する。当然ながら、分割も行われていない。つまり、分割階層レベルはゼロである。したがって、上記の分割条件は成立し、空間分割が行われる。上記の４次元空間に対して、ｘ軸、ｙ軸、ｚ軸、ｔ軸それぞれについて２分割を行うと、以下のサブ時空間が生成される。

【００４５】

サブ時空間０：０≦ｘ≦２，０≦ｙ≦２，０≦ｚ≦２，０≦ｔ≦２

サブ時空間１：－２≦ｘ≦０，０≦ｙ≦２，０≦ｚ≦２，０≦ｔ≦２

サブ時空間２：０≦ｘ≦２，－２≦ｙ≦０，０≦ｚ≦２，０≦ｔ≦２

サブ時空間３：－２≦ｘ≦０，－２≦ｙ≦０，０≦ｚ≦２，０≦ｔ≦２

サブ時空間４：０≦ｘ≦２，０≦ｙ≦２，－２≦ｚ≦０，０≦ｔ≦２

サブ時空間５：－２≦ｘ≦０，０≦ｙ≦２，－２≦ｚ≦０，０≦ｔ≦２

サブ時空間６：０≦ｘ≦２，－２≦ｙ≦０，－２≦ｚ≦０，０≦ｔ≦２

サブ時空間７：－２≦ｘ≦０，－２≦ｙ≦０，－２≦ｚ≦０，０≦ｔ≦２

サブ時空間８：０≦ｘ≦２，０≦ｙ≦２，０≦ｚ≦２，－２≦ｔ≦０

サブ時空間９：－２≦ｘ≦０，０≦ｙ≦２，０≦ｚ≦２，－２≦ｔ≦０

サブ時空間１０：０≦ｘ≦２，－２≦ｙ≦０，０≦ｚ≦２，－２≦ｔ≦０

サブ時空間１１：－２≦ｘ≦０，－２≦ｙ≦０，０≦ｚ≦２，－２≦ｔ≦０

サブ時空間１２：０≦ｘ≦２，０≦ｙ≦２，－２≦ｚ≦０，－２≦ｔ≦０

サブ時空間１３：－２≦ｘ≦０，０≦ｙ≦２，－２≦ｚ≦０，－２≦ｔ≦０

サブ時空間１４：０≦ｘ≦２，－２≦ｙ≦０，－２≦ｚ≦０，－２≦ｔ≦０

サブ時空間１５：－２≦ｘ≦０，－２≦ｙ≦０，－２≦ｚ≦０，－２≦ｔ≦０

生成されたサブ時空間０乃至１５のうち、次のサブ時空間がオブジェクト（obj0, obj1, obj2, obj3）のうちの少なくとも１つを包含する。

【００４６】

サブ時空間０：obj０，obj１

サブ時空間１：obj１

サブ時空間２：obj１

サブ時空間３：obj１

サブ時空間４：obj０，obj１，obj２

サブ時空間５：obj１

サブ時空間６：obj１

サブ時空間７：obj１，obj３

サブ時空間１５：obj３

上記から明らかなように、サブ時空間４は、３つのオブジェクトを含んでおり、かつ分割階層レベルは“１”である。すなわち、サブ時空間４に関しては、分割条件が成立する。したがって、サブ時空間４については、さらに空間分割が実行される。サブ時空間４に対して、ｘ軸、ｙ軸、ｚ軸、ｔ軸それぞれについて２分割を行うと、以下のサブ時空間がさらに生成される。

【００４７】

サブ時空間４－０：１≦ｘ≦２，１≦ｙ≦２，－１≦ｚ≦０，１≦ｔ≦２

サブ時空間４－１：０≦ｘ≦１，１≦ｙ≦２，－１≦ｚ≦０，１≦ｔ≦２

サブ時空間４－２：１≦ｘ≦２，０≦ｙ≦１，－１≦ｚ≦０，１≦ｔ≦２

サブ時空間４－３：０≦ｘ≦１，０≦ｙ≦１，－１≦ｚ≦０，１≦ｔ≦２

サブ時空間４－４：１≦ｘ≦２，１≦ｙ≦２，－２≦ｚ≦－１，１≦ｔ≦２

サブ時空間４－５：０≦ｘ≦１，１≦ｙ≦２，－２≦ｚ≦－１，１≦ｔ≦２

サブ時空間４－６：１≦ｘ≦２，０≦ｙ≦１，－２≦ｚ≦－１，１≦ｔ≦２

サブ時空間４－７：０≦ｘ≦１，０≦ｙ≦１，－２≦ｚ≦－１，１≦ｔ≦２

サブ時空間４－８：１≦ｘ≦２，１≦ｙ≦２，－１≦ｚ≦０，０≦ｔ≦１

サブ時空間４－９：０≦ｘ≦１，１≦ｙ≦２，－１≦ｚ≦０，０≦ｔ≦１

サブ時空間４－１０：１≦ｘ≦２，０≦ｙ≦１，－１≦ｚ≦０，０≦ｔ≦１

サブ時空間４－１１：０≦ｘ≦１，０≦ｙ≦１，－１≦ｚ≦０，０≦ｔ≦１

サブ時空間４－１２：１≦ｘ≦２，１≦ｙ≦２，－２≦ｚ≦－１，０≦ｔ≦１

サブ時空間４－１３：０≦ｘ≦１，１≦ｙ≦２，－２≦ｚ≦－１，０≦ｔ≦１

サブ時空間４－１４：１≦ｘ≦２，０≦ｙ≦１，－２≦

ｚ≦－１，０≦ｔ≦１
サブ時空間４－１５：０≦ｘ≦１，０≦ｙ≦１，－２≦ｚ≦－１，０≦ｔ≦１
生成されたサブ時空間４－０乃至４－１５のうち、次のサブ時空間がオブジェクト（obj0, obj1, obj2, obj3）のうち少なくとも１つを包含する。

【００４８】
サブ時空間４－０：ｏｂｊ０
サブ時空間４－４：ｏｂｊ２
サブ時空間４－８：ｏｂｊ０
サブ時空間４－１１：ｏｂｊ１
サブ時空間４－１２：ｏｂｊ２
上記から明らかなように、サブ時空間４－０，４－４，４－８，４－１１，４－１２のいずれも、１つのオブジェクトのみを包含する。したがって、分割条件は成立せず、ここで空間分割は終了する。

【００４９】次に、生成されたサブ時空間を用いて、レンダリングによる画像処理が実行される（ステップＳ２０３）。レンダリングに用いるアルゴリズムは、レイトレーシングであるとする。図６に例示するように、視点とスクリーンが定義される。視点およびスクリーンは、時間と共に変化しないものとする。ここでは、次のように視点とスクリーンが定義される。スクリーンは、４つの点で定義される。

【００５０】（１）視点
（ｘ，ｙ，ｚ）＝（１，１，４）
（２）スクリーン
（ｘ，ｙ，ｚ）＝（１．１，１．１，３）
（ｘ，ｙ，ｚ）＝（１．１，０．９，３）
（ｘ，ｙ，ｚ）＝（０．９，０．９，３）
（ｘ，ｙ，ｚ）＝（０．９，１．１，３）
次に、定義されたスクリーン上に１点（以下、「Ａ点」と呼ぶ）が、さらに定義される。そして、視点からこのＡ点を通る直線が、通過するサブ時空間が特定される。たとえばＡ点が、次の座標を有するとする。

【００５１】Ａ点座標：（ｘ，ｙ，ｚ）＝（１．０１，１．０１，３）
視点とＡ点を結ぶ直線は、サブ時空間０、サブ時空間４、サブ時空間８およびサブ時空間１２を通過する。したがって、Ａ点に対応する画素の処理が割り当てられたプロセッサコアは、サブ時空間０、サブ時空間４、サブ時空間８およびサブ時空間１２のオブジェクトデータを主メモリ等から読み込み、従属するローカルメモリに格納する。そして、プロセッサコアは、ローカルメモリに格納されたオブジェクトデータを用いて、Ａ点に対応する画素の描画処理を実行する。

【００５２】ここで、上記のサブ時空間は、時間情報を付加したオブジェクト情報に基づいて空間分割されたものである。したがって、視点とＡ点を結ぶ直線が通過する、サブ時空間０、サブ時空間４、サブ時空間８およびサブ時空間１２は、時間に応じて使用される。すなわち、－２≦ｔ≦０では、サブ時空間８およびサブ時空間１２が使用され、０≦ｔ≦２では、サブ時空間０およびサブ時空間４が使用される。たとえば時間単位“１”を６０フレームとすれば、－２≦ｔ≦０（６０×２＝１２０フレーム）では、サブ時空間８および１２のオブジェクトデータの転送が行われ、０≦ｔ≦２（６０×２＝１２０フレーム）では、サブ時空間０および４のオブジェクトデータの転送が実行される。したがって、ローカルメモリへのデータ転送回数は高々４回である。一方、フレームごとに空間分割をやり直し、主メモリ等からローカルメモリにサブ空間情報を転送し直すと、データ転送回数は、非常に大きいものとなる。つまり、サブ時空間０および８がサブ空間０に相当し、サブ時空間４および１２がサブ空間４に相当する場合、－２≦ｔ≦０（１２０フレーム）において、サブ空間０および４のオブジェクトデータの転送が１２０フレームそれぞれについて実行され、０≦ｔ≦２（１２０フレーム）においても、サブ空間０および４のオブジェクトデータの転送が１２０フレームそれぞれについて実行される。その結果、ローカルメモリへのデータ転送回数は４８０回となる。

【００５３】以上説明したように、本発明の第２の実施の形態によれば、アニメーションを表示する場合において、主メモリとローカルメモリ間のデータ転送回数を低減することできる。それにより、画像処理の高速化を図ることが可能となる。また、本発明の第２の実施の形態では、次に使用されるサブ時空間を予測することができる。このため、各プロセッサに従属するローカルメモリへのサブ時空間情報のプリロードが可能となり、画像処理のより一層の高速化を実現できる。

【００５４】上記の第２の実施の形態では、レンダリングに用いるアルゴリズムとしてレイトレーシングを用いて説明したが、本発明はこれに限られるものではない。たとえばアルゴリズムとしてビームトレーシングを用いても良い。ビームトレーシングでは、スクリーンを複数のサブ・スクリーンに分割し、分割されたサブ・スクリーンを各プロセッサに割り当てる。各プロセッサは、割り当てられたサブ・スクリーンの画像処理を行う。図１０に、図５の階層構造の空間をビームトレーシングに適用した例を示す。この例では、スクリーンを８つのサブ・スクリーンに分割している。たとえばサブ・スクリーン３の処理には、１２個のサブ空間が必要とされる。

【００５５】ここで、図９のステップＳ２０３のレンダリングにビームトレーシングが用いられた場合、ビームが通過するサブ空間は次のようになる。視点およびスクリーンの座標は上記と同様である。サブ・スクリーンは、スクリーン上の４つの点（以下、「Ｂ点、Ｃ点、Ｄ点、Ｅ点」と呼ぶ）で定義される。たとえばＢ点、Ｃ点、Ｄ点およびＥ点が、次の座標を有するとする。

【００５６】

Ｂ点座標：（ｘ，ｙ，ｚ）＝（１．０１，１．０１，３）
Ｃ点座標：（ｘ，ｙ，ｚ）＝（１．０１，１．０２，３）
Ｄ点座標：（ｘ，ｙ，ｚ）＝（１．０２，１．０２，３）
Ｅ点座標：（ｘ，ｙ，ｚ）＝（１．０２，１．０１，３）
視点とＢ，Ｃ，ＤおよびＥ点とを結ぶビームは、サブ時空間０、サブ時空間４、サブ時空間８およびサブ時空間１２を通過する。

【００５７】本発明の第２の実施の形態では、画像の作成手法としてバウンディングボリュームを用いても良い。レイトレーシングやビームトレーシングでは、オブジェクトが複雑な形状を有する場合や、交差判定に膨大な計算時間を要する場合がある。バウンディングボリュームは、複雑な形状を有するオブジェクトを単純なオブジェクトで包み込み、その単純なオブジェクトに対して交差判定を行う。したがって、計算時間の短縮化が図られる。また、モーションブラー手法を利用しても良い。この手法においても、上述したサブ時空間の利用が可能である。このため、レンダリングを用いた場合と同様に、転送回数の低減を図ることができる。

【００５８】（第３の実施の形態）次に、本発明の第３の実施の形態について説明する。本発明の第３の実施の形態は、上記の第１および第２の実施の形態のプロセッサコアの構造に関する。上記のオクツリー等の探索木を用いて３次元オブジェクト空間を分割した場合、各プロセッサコアが画像処理する際に必要となるデータは、大別して２つに分類される。すなわち、３次元空間の階層構造に関するデータ（以下、「階層構造データ」と呼ぶ）と、オブジェクトに関するデータ（以下、「オブジェクトデータ」と呼ぶ）、である。第１および第２の実施の形態で述べたように、各プロセッサコアは、割り当てられた領域の画像処理に必要なデータを、それぞれに従属するローカルメモリに記憶する。本発明の第３の実施の形態に係るプロセッサコアは、上記の階層構造データとオブジェクトデータの両方を同一のローカルメモリに記憶せずに、論理的または物理的に独立した異なるローカルメモリに別々に記憶する。

【００５９】図１１（Ａ）および（Ｂ）は、本発明の第３の実施の形態に係るプロセッサの構成を示すブロック図である。図１１（Ａ）に示すプロセッサコア６４は、１つのデータキャッシュ６６を論理的に２つの領域６６ａおよび６６ｂに分割する。そして、各領域６６ａ，６６ｂに、階層構造データおよびオブジェクトデータそれぞれを記憶する。たとえば領域６６ａに階層構造データを記憶し、領域６６ｂにオブジェクトデータを記憶する。一方、図１１（Ｂ）のプロセッサコア６８は、階層構造データを記憶する階層構造データキャッシュ７０と、オブジェクトデータを記憶するオブジェクトデータキャッシュ７２と、を備える。このプロセッサコア６８では、キャッシュ内に格納するデータの特性によって、各キャッシュメモリを最適化できる。たとえばオブジェクトデータは読み出しアクセスだけで書き込みアクセスがないデータである。したがって、オブジェクトデータキャッシュ７２をたとえばスクラッチパッドで構成すれば良い。スクラッチパッドは、高速メモリで構成され、データの読み込みを高速に行うことができるものである。したがって、高速にオブジェクトデータの読み込みが可能となり、画像処理の高速化が図られる。

【００６０】次に、図１１（Ｂ）のオブジェクトデータキャッシュ７２について詳細に説明する。最初に、一般的なキャッシュの構造について簡単に説明する。キャッシュの構造には、フルアソシアティブ（full associative）方式とダイレクトマップ（direct map）方式がある。図１２（Ａ）および（Ｂ）に、一般的なキャッシュ構造を示す。図１２（Ａ）が、フルアソシアティブ方式のキャッシュ構造であり、図１２（Ｂ）が、ダイレクトマップ方式のキャッシュ構造である。図１２（Ａ）および（Ｂ）のキャッシュ構造のいずれにおいても、アドレスのタグとキャッシュ中のタグの比較を行ないながら、データ読み出しの処理が実行される。アドレスのタグとキャッシュ中のタグとが等しく、有効ビットが設定されていれば、読み出しはキャッシュをヒットする。そして、対応するラインのデータが読み出される。そうでない場合は、読み出しはキャッシュをミスし、主メモリから該当するデータがリフィルされ、その後、そのデータが読み出される。いずれのキャッシュ構造であっても、データの最小単位であるラインのサイズは固定されている。そして、データのヒットチェック、リフィル等は、ラインごとに実行される。

【００６１】図１３に、本発明の第３の実施の形態に係るオブジェクトデータキャッシュに格納されるオブジェクトデータ例を示す。画像処理などの分野においては、データは、オブジェクトと呼ばれる、ある程度大きな単位で、処理される場合がある。図１３に示したオブジェクトデータは、球体のオブジェクトに関するデータであり、球に関する半径や、座標、色などの情報の集合である。図１３に例示した、オブジェクトデータの特徴としては、次のものが挙げられる。

【００６２】（１）１つのオブジェクトに関するデータは、メモリ上にまとまって配置される。

【００６３】（２）データに対するアクセスは、１つのオブジェクトを単位として実行される。

【００６４】（３）不要なデータは、１つオブジェクトを単位として発生する。

【００６５】図１３に例示するように、１つのオブジェクトの半径、座標等の属性を示すデータの各アドレスは、そのオブジェクトの先頭アドレスからのオフセット

（以下、「オフセット」と呼ぶ）によって表現することができる。したがって、上述したタグの比較は、オブジェクトの先頭アドレスに対してのみ行なえば十分である。しかしながら、上記の図１２（Ａ）および（Ｂ）に示したキャッシュ構造では、同一のオブジェクトのデータであっても、メモリ・アドレスによって個々に管理されている。このため、オブジェクトデータの読み出しは、オブジェクト単位ではなく、キャッシュのライン単位で、上述したタグの比較が実行されてしまう。このことは、無用なタグ比較を招くこととなり、タグ比較に要するコストを増大させてしまう。

【００６６】また、オブジェクトデータの読み出しがミスした場合、メインメモリからそのデータが読み出され、キャッシュのデータ部分に格納される。ところが、このキャッシュの入れ替えは、実際にデータがアクセスされるまでは、実行されない。このため、同一のオブジェクトに関するデータであって、連続的に使用されるデータがミスした場合には、各データのリフィルは、各データアクセスごとに発生してしまう。その結果、演算処理とリフィル処理が、交互に発生し、システムのパフォーマンスを低下させてしまう。

【００６７】本発明の第３の実施の形態に係るオブジェクトデータキャッシュは、データの取り扱いを、キャッシュのラインごとではなく、オブジェクトごとに、行なうことを可能とする。それにより、ヒットチェックに要する時間を短縮し、ハードウェアのコストを減少させることができる。また、データを先読みすることで、システムのパフォーマンスを向上させることができる。さらに、不要なオブジェクトデータを一括消去し、データ使用の効率化を図ることもできる。

【００６８】図１４に、本発明の第３の実施の形態に係るオブジェクトキャッシュの構造を示す。図１４に示すように、本発明の第３の実施の形態に係るオブジェクトキャッシュ７４は、オブジェクトデータを保持するキャッシュメモリ７６と、キャッシュメモリ７６中に配置されたオブジェクトデータを示すキャッシュテーブル７８と、メインメモリ８０に対してデータ転送を指令するオブジェクトトランスファコントローラ８２と、を少なくとも備える。

【００６９】図１５（Ａ）に示すように、キャッシュメモリ７６は、各オブジェクトデータを格納するデータフィールドと、そのデータが有効か否かを示すvalid bitフィールドと、そのデータに対応する、キャッシュテーブル７８中のエントリＮｏ．を示すエントリＮｏ．フィールドと、からなる、複数のラインを有している。また、キャッシュテーブル７８は、上記の図１２（Ａ）および（Ｂ）に示したキャッシュ構造を採り、図１５（Ｂ）に示すように、各オブジェクトデータを格納する、キャッシュメモリ７６中のラインのスタートラインＮｏ．を示すスタートラインＮｏ．フィールドと、そのラインに格納されたオブジェクトの識別番号を示すオブジェクトＮｏ．フィールドと、そのスタートラインＮｏ．が有効か否かを示すvalid bitフィールドと、からなる、複数のエントリを有している。

【００７０】次に、本発明の第３の実施の形態に係るオブジェクトキャッシュの動作について図１４、図１５（Ａ）および（Ｂ）を用いて説明する。図１４、図１５（Ａ）および（Ｂ）では、データ読み出しの際には、ＣＰＵ８４は、オブジェクトの識別番号およびオフセットを、オブジェクトキャッシュ７４に指示するものとする。ＣＰＵ８４によるデータアクセス要求があった場合、そのデータがキャッシュメモリ７６内に存在すれば（オブジェクトヒット）、キャッシュテーブル７８から、そのデータに対応するスタートラインＮｏ．が出力される。出力されたスタートラインＮｏ．と、ＣＰＵ８４からのオフセットとは、加算器８６によって加算され、キャッシュメモリ７６に出力される。そして、キャッシュメモリ７６に格納されたオブジェクトデータがＣＰＵ８４に出力され、それと同時に、そのデータが有効であることを示すvalid信号もＣＰＵ８４に渡される。

【００７１】一方、ＣＰＵ８４によって要求されたデータがキャッシュメモリ７６中に存在しなければ（オブジェクトミス）、valid信号はＣＰＵ８４に出力されない。ＣＰＵ８４は、オブジェクト識別番号をオブジェクトトランスファコントローラ８２に渡し、オブジェクトトランスファコントローラ８２にキャッシュメモリ７６へのデータ転送を指示する。オブジェクトトランスファコントローラ８２は、受け取ったオブジェクト識別番号のオブジェクトデータすべてを、主メモリ８０から読み出し、キャッシュメモリ７６にリフィルする。このリフィルの際、オブジェクトの各データの大きさが一定であれば、主メモリ８０から一定量のデータが転送される。一方、オブジェクトの各データの大きさが異なる場合には、異なる大きさのデータが転送されることになる。このリフィルの作業は、オブジェクトトランスファコントローラ８２によって実行され、ＣＰＵ８４の動作とは独立に行われる。各データのリフィルが順次終了次第、ＣＰＵ８４にそのデータおよびvalid信号が出力される。また、その出力されたデータを用いた、ＣＰＵ８４の処理と並列して、その他のデータのリフィルも実行される。

【００７２】本発明の第３の実施の形態に係るオブジェクトキャッシュ７４では、ヒットチェックが、キャッシュメモリ７６の各ラインに対してではなく、キャッシュテーブル７８の各エントリに対して実行される。キャッシュテーブル７８のエントリの数は、キャッシュメモリ７６のラインの数と比べて小さいので、ヒットチェックのコストを低減できる。また、オブジェクトデータ全体がリフィルされるので、同一オブジェクトのデータが連続して利用される場合には、必要なデータのプリロード

が可能となる。このため、データリフィルのペナルティーを小さくできる。

【００７３】次に、本発明の第３の実施の形態に係るオブジェクトキャッシュの動作をより詳細に説明する。ここでは、図１６に示すように、図１４のキャッシュメモリ７６が、各ラインを配置可能な場所が２つ存在する、２ウェイセットアソシアティブ方式を採り、一方、キャッシュテーブル７８が、各エントリの配置場所が固定されない、フルアソシアティブ方式を採用する。キャッシュテーブル７８は、ＣＡＭ（Content Addressable Memory）を用いて、キャッシュメモリ７６の各ラインを求める。なお、以下では、オブジェクトヒットの場合とオブジェクトミスの場合を分けて説明するが、実際には、これらは並列処理される。

【００７４】初期状態では、キャッシュメモリ７６およびキャッシュテーブル７８のすべてのvalid bitは、ゼロである。そして、ＣＰＵ８４によるデータアクセス要求があると、キャッシュテーブル７８は、ＣＰＵ８４からオブジェクト識別番号を受け取る。オブジェクト識別暗号を受け取ったキャッシュテーブル７８は、次のものを出力する。

【００７５】（１）入力されたオブジェクト識別番号が、キャッシュテーブル７８中に存在することを示すヒット信号

（２）入力されたオブジェクト識別信号のオブジェクトの各データを格納する、キャッシュメモリ７６中のラインのスタートラインＮｏ．

（３）スタートラインＮｏ．が有効であることを示すvalid信号

（４）キャッシュテーブル７８中のいずれのエントリがヒットしたかを示すヒットエントリＮｏ．

ここで、上記の（２）乃至（４）は、上記の（１）がセットされている場合のみに有効である。

【００７６】オブジェクトヒットの場合、上記の（１）はセットされるので、上記の（２）乃至（４）は有効となる。そして、加算器８６によって、（２）のスタートラインＮｏ．とオフセットが加算される。キャッシュメモリ７６へのアクセスは、その加算された結果であるアドレスに対して、行われる。アクセスされると、キャッシュメモリ７６は、各ウェイから、次の信号を出力する。

【００７７】（５）アクセスされたラインが有効であることを示すvalid信号

（６）そのラインがキャッシュテーブル７８のどのエントリに対応するかを示すエントリＮｏ．

（７）そのラインに格納されたオブジェクトデータ

ここで、上記の（６）および（７）は、上記の（５）がセットされている場合のみに有効である。

【００７８】オブジェクトヒットの場合、いずれかのウェイでは、上記の（４）のヒットエントリＮｏ．と（６）のエントリＮｏ．とが一致する。そして、一致したウェイの（５）のvalid信号および（７）のオブジェクトデータが選択される。選択されたオブジェクトデータはＣＰＵ８４に渡される。一方、ＣＰＵ８４に出力されるvalid信号は、上記の（１）、（３）、（４）と各ウェイの（６）との比較結果の論理和、および選択された（５）、の論理積となる。

【００７９】次に、オブジェクトミスの場合について図１７を用いて説明する。オブジェクト識別番号を受け取ったキャッシュテーブル７８は、要求されたオブジェクトのデータが、キャッシュメモリ７６中に存在しないことを示すミス信号を、オブジェクトトランスファコントローラ８２に、出力する。ミス信号を受け取ったオブジェクトトランスファコントローラ８２は、データ転送可能状態であれば、キャッシュテーブル７８中のいずれのエントリに書き込みするかを示す情報（refillentry No.）を、内蔵されたレジスタ８８に保持する。この情報は、オブジェクトデータの転送が終了するまで保持される。

【００８０】また、オブジェクトトランスファコントローラ８２は、オブジェクト識別番号を取得し、主メモリ８０中における、そのオブジェクトデータの格納場所を特定する。主メモリ８０中に、そのオブジェクトデータの開始アドレスやデータサイズが保持されている場合には、オブジェクトトランスファコントローラ８２は、それらを取得し、データ転送に備える。

【００８１】データ転送開始時、オブジェクトトランスファコントローラ８２は、転送開始信号（transfer start）と、キャッシュメモリ７６中のスタートラインＮｏ．を、キャッシュテーブル７８に、出力する。キャッシュテーブル７８は、受け取ったスタートラインＮｏ．を、指定されたエントリのスタートラインＮｏ．フィールドに書き込む。そして、その書き込みを行なったエントリのvalid bitフィールドをvalid bitをセットする。このvalid bitは、各オブジェクトのデータ転送開始時に、順次セットされる。このため、各データがリフィルされ次第、順次ＣＰＵ８４に転送が可能となる。

【００８２】また、オブジェクトトランスファコントローラ８２は、書き込み信号（writeenable）、書き込みが行われるウェイ番号、書き込みが行われる、キャッシュメモリ７６中のラインＮｏ．、を出力する。書き込みが行われるラインＮｏ．は、オブジェクトトランスファコントローラ８２が管理する。具体的には、ラインＮｏ．は、スタートラインＮｏ．から順に、書き込みが行われる度に、１ずつカウントアップさせることで、決定される。オブジェクトトランスファコントローラ８２は、キャッシュメモリ７６の各ラインのエントリＮｏ．フィールドに、そのラインに対応するキャッシュテーブル７８中のエントリＮｏ．を書き込み、データフィールドに、主メモリから取得したオブジェクトデータを、書

き込む。ウェイの選択は、ＬＲＵ法などによって行なえば良い。また、転送されるオブジェクトデータによっては、再利用の可能性が、ＣＰＵ８４の演算結果から求められる場合がある。この場合には、オブジェクトトランスファコントローラ８２が、ＣＰＵ８４から情報を得ることで、ウェイの選択を実行できる。

【００８３】キャッシュメモリ７６は、データ書き込みと同時に、各ラインのvalid bitをセットする。すでにvalid bitがセットされているラインについては、そのラインのエントリＮｏ．フィールドに格納されたエントリＮｏ．をキャッシュテーブル７８に出力する。キャッシュテーブル７８は、受け取ったエントリＮｏ．のエントリのvalid bitをリセットし、そのエントリを無効にする。

【００８４】次に、本発明の第３の実施の形態に係るキャッシュメモリ７６およびキャッシュテーブル７８に対する書き込みが、どのようにして行われるかについて図１８、図１９（ａ）乃至（ｅ）を用いて説明する。図１８は、主メモリ８０中に格納された複数のオブジェクトのデータの例である。図１９（ａ）乃至（ｅ）は、図１８のデータがキャッシュメモリ７６に書き込まれる際の、キャッシュメモリ７６およびキャッシュテーブル７８の内容の変化を示す図である。説明の簡単化を図るため、図１８の各オブジェクトは、オブジェクト１、オブジェクト２、オブジェクト３、…、オブジェクト８の順でＣＰＵ８４からアクセスされるものとする。また、図１９（ａ）乃至（ｅ）では、キャッシュメモリ７６が、１ウェイ１６ラインであり、キャッシュテーブル７８が、４エントリであるとする。

【００８５】まず、初期化状態では、キャッシュメモリ７６およびキャッシュテーブル７８のすべてのvalid bitは、リセットされている。この場合、ＣＰＵ８４がオブジェクト１に対してアクセス要求しても、キャッシュテーブル７８中には、オブジェクト１に対応するエントリは存在しない（図１９（ａ）参照）。したがって、キャッシュメモリ７６の０ラインから順に、オブジェクト１のデータがリフィルされる。また、同時に、このリフィルが、キャッシュテーブル７８の０エントリに対応することも、書き込まれる。一方、キャッシュテーブル７８には、書き込まれたデータがオブジェクト１のものであることと、そのデータが書き込まれたラインのスタートラインＮｏ．と、が書き込まれる（図１９（ｂ）参照）。ＣＰＵ８４がオブジェクト２に対してアクセス要求した場合も同様に行われる（図１９（ｃ）参照）。

【００８６】次に、ＣＰＵ８４がオブジェクト３に対してアクセス要求した場合、キャッシュテーブル７８の２エントリにオブジェクト３に関する情報が書き込まれ、キャッシュメモリ７６のライン１２から順にオブジェクトデータが書き込まれる（図１９（ｄ）参照）。キャッシュメモリ７６の１５ラインまで書き込まれると、再び０ラインから書き込みが行われる。この時、０ラインおよび１ラインに対応する、キャッシュテーブル７８の０エントリのvalid bitは、リセットされる（図１９（ｅ）参照）。

【００８７】（第４の実施の形態）次に、本発明の第４の実施の形態について説明する。複数のノードと１つのターゲットがネットワークで相互に接続されているシステムにおいて、複数のノードからターゲットに対して同時にアクセス要求が起きた場合、そのアクセス要求の競合によってシステム全体の性能が低下してしまう場合がある。この性能低下は、図７に示した第１の実施の形態の画像処理システムにおいても、同様に起こり得る。すなわち、図７のチップ５２、ボード５６、コンピュータ５８それぞれの内部においても、アクセス要求の競合が起こり得る。複数のコンピュータ５８を接続するネットワーク６０においても、もちろん同様である。本発明の第４の実施の形態は、複数のノードから同時に１つのターゲットにアクセス要求が頻繁に生じる画像処理システムに係るものである。すなわち、同一のターゲットに対する、複数のアクセス要求を１つにまとめてバースト転送を行うことで、各ノードからターゲットへのアクセス待ち時間（レイテンシ）を小さくし、システム全体の性能低下を防止するものである。

【００８８】以下、本発明の第４の実施の形態について図２０に示すシステム構成を用いて説明する。図２０は、単一の共有メモリ９０とｎ個のプロセッサ９２－０，９２－１，…，９２－ｎ－１とが、ネットワーク９４を介して接続された、共有メモリ型マルチプロセッサを示している。もちろん、本発明の適用範囲は、マルチプロセッサに限られるものではない。まず最初に、本発明の第４の実施の形態に係るコンバイン機能（Combining機能）および拡張コンバイン機能（拡張Combining機能）について説明する。

【００８９】（コンバイン機能）図２１は、図２０のプロセッサ９２－０からのアクセス要求とプロセッサ９２－１からのアクセス要求の両方が、共有メモリ９０の「0x1000」番地への読み込みアクセスである場合を示している。この場合、この２つのアクセス要求を１つにまとめ、１つのアクセス要求として取り扱うことで、アクセス競合は避けられる。この２つのアクセス要求を１つにまとめて取り扱う機能を「コンバイン機能」と呼ぶ。このコンバイン機能によって、２つのプロセッサ９２－０，９２－１からの読み込みアクセスは１つにまとめられ、その１つの読み込みアクセスに基づいて、共有メモリ９０の「0x1000」番地のデータが読み出される。そして、その読み出されたデータは２つのプロセッサ９２－０，９２－１それぞれに送られる。このようにコンバイン機能は、同一のアドレスに対するアクセス要求を複数のプロセッサが同時に行った場合に、そのアクセス要求を１つにまとめることで、アクセス要求の競合を回避

し、それにより各プロセッサのアクセス待ち時間を低減できる。

【００９０】（拡張コンバイン機能）図２２（Ａ）は、図２０のプロセッサ９２－０からの１ワード（４バイト）アクセス要求が、共有メモリ９０の「0x1000」番地への読み込みアクセス、プロセッサ９２－１からの１ワードアクセス要求が、「0x1004」番地への読み込むアクセス、である場合を示している。この場合、２つの読み込み先アドレスが同じではないため、上記のコンバイン機能によっては２つのアクセス要求を１つにまとめることはできない。拡張コンバインは、上記のコンバイン機能とは異なり、図２２（Ｂ）に示すように、異なるアドレスに対する複数のアクセス要求を、１つにまとめることができるものである。この拡張コンバイン機能は、各ノードからの複数のアクセス要求が互いに近接するアドレスに対して行われることが多いシステムにおいて非常に有効なものである。もちろん、同一アドレスに対する複数のアクセス要求の場合であっても構わない。

【００９１】次に、本発明の第４の実施の形態に係る待ち行列（queue）システムについて説明する。本発明の第４の実施の形態に係る待ち行列システムは、上記の拡張コンバイン機能を取り入れたシステムである。図２３に、本発明の第４の実施の形態に係る待ち行列システムの構成を示す。この待ち行列システムは、図２０に示したマルチプロセッサに適用された例である。図２３に示すように、本発明の第４の実施の形態に係る待ち行列システムは、調停部９６と、比較部９８と、待ち行列部１００と、マルチキャスト部１０２と、を少なくとも備える。以下、それぞれについて説明する。

【００９２】（調停部）ＬＳＩ内部では、高い動作周波数が要求される。このため、図２０のｎ個のプロセッサ９２－０，…，９２－ｎ－１すべてから同時に共有メモリ９０にアクセス要求があった場合、１クロック・サイクル内で、それらすべてのアクセス要求を比較し、上記の拡張コンバインを実行することは現実的には困難である。図２３の調停部９６は、複数のアクセス要求の中から、比較部９８に送られるアクセス要求を、１クロック・サイクルごとに決定する。図２４に、調停部９６の構成を示す。図２４に示すように、この調停部９６は、マルチプレクサ１０４と、アクセステーブル１０６と、を少なくとも備える。マルチプレクサ１０４は、複数のプロセッサ９２－０，…，９２－ｎ－１からのアクセス要求の中から幾つかを選び出し、比較部９８に送る。マルチプレクサ１０４は、最大ｎ個のアクセス要求の中から、最大ｋ本のアクセス要求を選択できる。アクセステーブル１０６は、アクセス要求が待ち行列部１００に格納されているプロセッサを示す、複数のビットを有している。各ビットは、プロセッサ９２－０，…，９２－ｎ－１それぞれに対応する。そして、各ビットは、待ち行列部１００にアクセス要求が格納されると、“１”にセットされ、そのアクセス要求が実行されるまで“１”を保持する。マルチプレクサ１０４は、アクセステーブル１０６を参照して、比較部９８に送るアクセス要求を選択する。

【００９３】ここで、待ち行列部１００がブロッキング（blocking）である場合、調停部９６は、待ち行列部１００にアクセス要求がすでに格納されているプロセッサからの新たなアクセス要求は選択しない。一方、ノンブロッキング（nonblocking）な待ち行列部１００の場合、アクセステーブル１０６に各プロセッサからのアクセス要求を過去何サイクル前に受け付けたか、あるいは各プロセッサからのアクセス要求を過去幾つ受け付けたかを示せばよい。アクセステーブル１０６を参照してアクセス要求を選択すれば、各プロセッサからのアクセス要求をより公平に調停（アービトレーション）することが可能となる。なお、この場合には、アクセステーブル１０６は、各プロセッサに複数のビットを割り当てて、各プロセッサからのアクセス要求の状態を示すことになる。

【００９４】（比較部）比較部９８は、調停部９６によって選択されたアクセス要求と待ち行列部１００の各エントリに格納されたアクセス要求とを比較する。図２５に、比較部９８の構成を示す。比較部９８は、まず最初に、調停部９６によって選択されたアクセス要求と待ち行列部１００の先頭エントリに格納されているアクセス要求と、を比較する。この２つのアクセス要求が拡張コンバイン可能であれば、比較部９８は、調停部９６で選択されたアクセス要求を待ち行列部１００の先頭エントリにそのまま送る。そして、送られたアクセス要求は、先頭エントリに格納されているアクセス要求と拡張コンバインされる。一方、拡張コンバイン不可能である場合には、調停部９６によって選択されたアクセス要求を、待ち行列部１００の第２番目のエントリに格納されているアクセス要求と比較する。この２つのアクセス要求が拡張コンバイン可能であれば、待ち行列部１００に送る。一方、不可能である場合には、待ち行列部１００の第３番目のエントリに格納されているアクセス要求と比較する。以下同様にして比較を行い、待ち行列部１００のすべてのエントリに格納されているアクセス要求と拡張コンバイン不可能である場合には、調停部９６によって選択されたアクセス要求を、待ち行列部１００の最後尾のエントリに格納する。これらの比較処理は、処理速度向上のため、パイプライン処理で実行される。

【００９５】調停部９６は、複数のプロセッサからのアクセス要求の中から幾つかを選び出し、比較部９８に出力する。このため、拡張コンバインの可能性が低下してしまうおそれがある。本発明の第４の実施の形態では、調停部９６および比較部９８の動作周波数をその他の動作周波数の逓倍とすることで、一度に受け付けるアクセス要求の数の増大を図っている。このことは、各部が互

いに独立動作していることに基づくものである。

【００９６】（待ち行列部）待ち行列部１００は、複数のエントリを有しており、各エントリにプロセッサからのアクセス要求を格納する。各エントリに格納されたアクセス要求は、先に格納された順に待ち行列部１００から取り出される。図２６に、待ち行列部１００の各エントリの構成を示す。各エントリは、アクセス要求が読み込み動作であるか書き込み動作であるかを示す、１ビット、アクセス要求のアクセスタイプ（たとえば、バーストモード、ワードモード、ハーフモードおよびバイトモードの４種類）を示す、２ビット、ターゲットアドレスを示す、そのアドレス幅分だけの、ビット、を備える。また、各エントリは、拡張コンバインの結果を記憶するコンバイン情報テーブルを有する。このテーブルの実施方法は種々あるが、たとえば８つのプロセッサから構成され、最大バースト数が８であるシステムの場合、バーストアクセスに関するテーブルは、図２７に示すテーブルとなる。このテーブルは、プロセッサ０からアドレス「0x1010」への４word-Burst-readと、プロセッサ１からアドレス「0x1008」への２word-Burst-readと、プロセッサ２およびプロセッサ３からアドレス「0x1000」への２word-Burst-readと、が拡張コンバインされていることを示している。また、ワード／ハーフ／バイトアクセスに関するテーブルは、図２８に示すテーブルとなる。このテーブルはプロセッサ２およびプロセッサ５からアドレス「0x1000」へのByte-readと、プロセッサ４からアドレス「0x1002」へのByte-Readと、が拡張コンバインされていることを示している。

【００９７】ここで、待ち行列部１００は、書き込み動作時のデータを格納するためのバッファメモリを備えている。このバッファメモリは、スタック構造を採用し、その構造に追加された最新のデータを最初に使用するようになっている。そのため、待ち行列部１００の各エントリそれぞれに対応してメモリ領域を確保する必要はない。したがって、メモリコスト低減の点から望ましいものである。また、このバッファメモリはスタックポインタを内蔵する。スタックポインタ１、２、３はバッファメモリのどこからデータを取り出すかを指示するレジスタである。具体的にはどのエントリの書き込み要求についてのデータであるかを指示している。図２９のバッファメモリでは一番下にあるデータ１が現在アクセス中若しくは最も近未来にアクセスされるデータである。

【００９８】（第５の実施の形態）次に、本発明の第５の実施の形態について説明する。レイトレーシングは、各画素ごとにオブジェクトとの交差判定を行ない、交差するオブジェクトの彩度や明度を計算する。このため、交差判定対象のオブジェクトデータの総量が、プロセッサに従属するローカルメモリの容量に比べて大きいと、ローカルメモリに対するオブジェクトデータの入れ替えが頻繁に発生することになる。本発明の第５の実施の形態は、各オブジェクトごとに各画素の交差判定を行うことで、主メモリから各プロセッサのローカルメモリへのオブジェクトデータの転送を最小限に抑え、それにより画像処理の高速化を図るものである。ここでは、各プロセッサには、画面全体の中の４×４（＝１６）画素が割り当てられている。また、各プロセッサは、割り当てられた画素の処理の途中の状態を保存するステータステーブルを有している。このステータステーブルは、画素の数だけのエントリを有してる。１６個の画素を処理するプロセッサのステータステーブルは、１６個のエントリを有することになる。各エントリは、オブジェクト識別番号、交点座標、および、視点から交点までの距離、それぞれを格納する３つのフィールドを備える。オブジェクト識別暗号に換えて、オブジェクトデータを格納する主メモリ上の開始アドレスとしても良い。なお、以下では、それぞれを、オブジェクトフィールド、交点座標フィールド、距離フィールドと呼ぶ。

【００９９】本発明の第５の実施の形態の動作は次の通りである。図３０は、本発明の第５の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。最初に、ステータステーブルが初期化される。すなわち、ステータステーブルの各エントリのオブジェクトフィールド、交点座標フィールドは初期化され、距離フィールドは無限大に設定される（ステップＳ３０１）。任意のオブジェクトが選択され、そのオブジェクトデータが主メモリからローカルメモリに読み込まれる（ステップＳ３０２）。次に、１つの画素が選択され（ステップＳ３０３）、その画素を通る光線と上記のステップＳ３０２で選択されたオブジェクトとの交差判定が行われる。交差する場合には、視点からその交点までの距離が求められる。そして、その求められた結果が、距離フィールドの値よりも小さければ、オブジェクトフィールド、交点座標フィールドおよび距離フィールドの書き換えが行われる（ステップＳ３０４）。すべての画素について上記のステップＳ３０４が実行され（ステップＳ３０５ＹＥＳ）、すべてのオブジェクトについて上記のステップＳ３０３およびステップＳ３０４が実行されると（ステップＳ３０６ＹＥＳ）、この動作は終了する。

【０１００】この動作終了後、各画素に対応するエントリには、交差判定の結果が示されている。すなわち、いずれのオブジェクトとも交差しない画素の距離フィールドは無限大のままである。一方、いずれかのオブジェクトと交差する画素のオブジェクトフィールドは、その交差するオブジェクト識別番号を、交点座標フィールドは、その交点座標を、距離フィールドは、視点からその交点までの距離を、それぞれ格納しているはずである。

【０１０１】本発明の第５の実施の形態によれば、主メモリからローカルメモリへのオブジェクトデータの転送は、そのオブジェクトの数だけ行えばよいことになる。したがって、転送回数は大幅に低減され、その結果、画

像処理の高速化が図られることになる。なお、オブジェクトデータをスクリーン上に透視変換することで、各オブジェクトについて交差判定の不要な画素を予め特定することが可能である。この場合には、交差判定自体の回数も低減され、より画像処理の高速化が促進されることになる。

【０１０２】次に、処理対象の３次元空間が上記のオクツリーにより空間分割されている場合における、本発明の第５の実施の形態について説明する。この場合には、各プロセッサのステータステーブルの各エントリには、上記のオブジェクトフィールド、交点座標フィールド、距離フィールドに加えて、対応する画素の処理が終了したか否かを示す終了フィールド、次に処理されるサブ空間を示すサブ空間フィールド、をさらに備える。この場合の動作は次の通りである。図３１は、本発明の第５の実施の形態に係る画像処理方法の処理手順を示すフローチャートであって、処理対象の３次元空間がオクツリーで空間分割されている場合を示すフローチャートである。まず最初に、ステータステーブルが初期化される。すなわち、各エントリのオブジェクトフィールド、交点座標フィールド、サブ空間フィールドが初期化され、終了フィールドが未終了状態、距離フィールドが無限大に設定される（ステップＳ４０１）。終了フィールドが未終了状態である画素すべてが選択され（ステップＳ４０２）、各画素に対応する光線が次に通過するサブ空間が特定される。その特定されたサブ空間の番号が各画素に対応するエントリのサブ空間フィールドに書き込まれる。ただし、光線が通過するサブ空間の存在しない画素については、その終了フィールドに処理終了が書き込まれる（ステップＳ４０３）。

【０１０３】次に、終了フィールドが未終了状態である画素が１つ選択され（ステップＳ４０４）、そのサブ空間フィールドが示すサブ空間に存在するオブジェクトデータが主メモリからローカルメモリに読み込まれる（ステップＳ４０５）。そして、選択された画素と読み込まれたオブジェクトとの交差判定が行われる。さらに、そのサブ空間をサブ空間フィールドに示す他の画素がある場合には、その画素についても同様に交差判定が行われる。交差する場合には、視点からその交点までの距離が求められる。そして、その求められた距離が、距離フィールドの値よりも小さければ、オブジェクトフィールド、交点座標フィールドおよび距離フィールドの書き換えが行われる。そして、終了フィールドにその画素の処理が終了したことが書き込まれる。ステップＳ４０５で読み込まれたオブジェクトデータが複数ある場合には、すべてのオブジェクトについて同様に行われる（ステップＳ４０６）。すべての画素の処理が終了すると（ステップＳ４０７ＹＥＳ）、この動作は終了する。

【０１０４】この動作終了後、各画素のエントリには、交差判定の結果が示される。すなわち、いずれのオブジェクトとも交差しない画素の距離フィールドは無限大のままである。一方、いずれかのオブジェクトと交差する画素のオブジェクトフィールドは、その交差するオブジェクトの番号を、交点座標フィールドは、その交点座標を、距離フィールドは、視点からその交点までの距離を、それぞれ格納しているはずである。

【０１０５】次に、上記のステータステーブルに対する書き込みが、どのように行われるかについて図３２、図３３（ａ）乃至図３４（ｇ）を用いて説明する。ここでは、図３２に示す２次元平面内のオブジェクトとの交差判定を行う場合について説明する。また、図３３（ａ）乃至図３４（ｇ）は、図３２のオブジェクトとの交差判定における、ステータステーブルの変化を示す図である。図３２では、説明の簡単化を図るため、処理対象の空間を２次元で説明するが、処理対象が３次元空間であっても何ら異なる点はない。図３２に示すように、この２次元空間１０８内には、６つのオブジェクトobj1, obj2, obj3, obj4, obj5, obj6が存在している。そして２次元空間１０８は、サブ空間S1, S2, S3, S4, S5, S6, S7, S8, S9, S10, S11, S12, S13に分割されている。スクリーン１１０上には、４つの画素ｐ０，ｐ１，ｐ２，ｐ３がある。

【０１０６】ステータステーブルの各エントリが初期化される。具体的には、終了フィールドは、未終了状態（No）に、距離フィールドは、無限大（∞）に、それぞれ設定される。また、その他のフィールド（サブ空間フィールド、オブジェクトフィールド、交点座標フィールド）は、リセットされる（図３３（ａ）参照）。

【０１０７】最初、すべての画素ｐ０乃至ｐ３は、処理が未終了である。したがって、すべての画素が選択される。そして、各画素を通過する光線が、次に通過するサブ空間が特定される。特定されたサブ空間が、各画素に対応するサブ空間フィールドに、書き込まれる。図３２では、各画素を通過する光線はすべて、最初、サブ空間Ｓ１２を通過する。したがって、各サブ空間フィールドには、「Ｓ１２」が書き込まれる（図３３（ｂ）参照）。

【０１０８】サブ空間Ｓ１２内のオブジェクトobj５、obj６のデータが、主メモリからローカルメモリに読み込まれる。そして、各画素の光線とobj５およびobj６との交差判定が行われる。図３２では、画素ｐ０の光線がobj５と交差し、画素ｐ２の光線がobj６と交差する。したがって、画素ｐ０のオブジェクトフィールドに、オブジェクトobj５が、交点座標フィールドに、交点座標（ｘ１，ｙ１）が、距離フィールドに、距離ｄ１が、それぞれ書き込まれる。同様に、画素ｐ２のオブジェクトフィールドに、オブジェクトobj６が、交点座標フィールドに、交点座標（ｘ２，ｙ２）が、距離フィールドに、距離ｄ２が、それぞれ書き込まれる。そして画素ｐ０およびｐ２の両方の終了フィールドに、処理が終了したことが書き込まれる。具体的には処理の終了を意味する「Ye

s」が、書き込まれる（図３３（ｃ）参照）。

【０１０９】未終了状態の画素ｐ１およびｐ３が選択される。そして、各画素の光線が、次に通過するサブ空間が特定される。図３２では、画素ｐ１およびｐ３の光線は、サブ空間Ｓ１２の次に、サブ空間Ｓ１３を通過する。画素ｐ１およびｐ３のサブ空間フィールドには、「Ｓ１３」が書き込まれる（図３３（ｄ）参照）。

【０１１０】サブ空間Ｓ１３内のオブジェクトobj６のデータが、主メモリからローカルメモリに読み込まれる。そして、画素ｐ１およびｐ３の光線とobj6との交差判定が行なわれる。図３２では、画素ｐ１およびｐ３の光線は共に、オブジェクトobj６とは交差しない。したがって、画素ｐ１およびｐ３それぞれのオブジェクトフィールド、交点座標フィールド、距離フィールドは、変化しない（図３３（ｄ）参照）。

【０１１１】再び、未終了状態の画素ｐ１およびｐ３が選択される。そして、各画素の光線が、次に通過するサブ空間が特定される。図３２では、画素ｐ１およびｐ３の光線は、サブ空間Ｓ１３の次に、サブ空間Ｓ１０を通過する。画素ｐ１およびｐ３のサブ空間フィールドには、「Ｓ１０」が書き込まれる（図３４（ｅ）参照）。

【０１１２】サブ空間Ｓ１０内のオブジェクトobj３のデータが、主メモリからローカルメモリに読み込まれる。そして、画素ｐ１およびｐ３の光線とobj３との交差判定が行なわれる。図３２では、画素ｐ１の光線は、obj3と交差する。画素ｐ１のオブジェクトフィールドに、オブジェクトobj３が、交点座標フィールドに、交点座標（ｘ３，ｙ３）が、距離フィールドに、距離ｄ３が、それぞれ書き込まれる。そして、終了フィールドに、「Yes」が、書き込まれる。一方、画素ｐ３の光線は、オブジェクトobj３とは交差しない。したがって、画素ｐ３のオブジェクトフィールド、交点座標フィールド、距離フィールドは、変化しない（図３４（ｆ）参照）。

【０１１３】以下、同様に処理が行なわれる。図３２では、画素ｐ３の光線は、いずれのサブ空間とは交差しない。したがって、画素ｐ３の終了フィールドに、「Yes」が書き込まれる。また、距離フィールドは、無限大（∞）のままとなる（図３４（ｇ）参照）。

【０１１４】以上説明したように、本発明の第５の実施の形態によれば、主メモリからローカルメモリへのデータ転送量を大幅に低減することができる。したがって、画像処理の高速化を図ることができる。たとえば、プロセッサに従属するローカルメモリの容量がｓ、３次元空間内のすべてのオブジェクトデータがローカルメモリ容量ｓのｒ倍、画素の数がｎの場合、転送されるデータ量は、ｓ・ｒ・ｎとなる。なぜなら、従来では、各画素ごとにすべてのオブジェクトデータを順にロードして交差判定を行い、順に輝度を求めていくからである。一方、本発明の第５の実施の形態では、すべてのオブジェクトデータは、一度だけ主メモリからローカルメモリに転送されるだけである。したがって、転送されるデータ量はｓ・ｒで済む。

【０１１５】さらに、空間分割によっても、各サブ空間に含まれるオブジェクトのデータが各プロセッサに従属するローカルメモリの容量よりも大きくなってしまう場合、同様に上記方法によりデータ転送量を大幅に低減できる。また、オブジェクトデータの先読みが可能なシステムにおいては、複数の画素の処理が終了するまでに次のオブジェクトデータの転送が終了していればよい。このため、本発明の第５の実施の形態によれば、従来と比べて転送のオーバーヘッドが生じにくい。１つの主メモリを有し、複数のプロセッサを用いて並列に画像処理を行う画像処理システムでは、アクセス要求の競合からデータ転送のオーバーヘッドが大きい。したがって、本発明の第５の実施の形態は、このようなシステムに非常に有効である。

【０１１６】（第６の実施の形態）次に、本発明の第６の実施の形態について説明する。アニメーション（動画）の１コマを均等な画素数の領域に分割し、各領域を複数のプロセッサで分担して画像処理する場合、各プロセッサに割り当てられた領域内の物体の数や形状の違いにより、各プロセッサの処理時間に大きな差が生じることがある。１コマ全体の描画に要する時間は、各プロセッサの描画処理の中で最も長いものに依存してしまう。このため、１つのプロセッサの処理時間が他と比べて極端に長いと、全体としては画像処理の遅延化を招いてしまう。また、先行して処理が終了したプロセッサは他のすべてのプロセッサの処理が終了するまでは何も処理しない状態となる。このことは、プロセッサの利用効率を低下させる。

【０１１７】本発明の第６の実施の形態は、各プロセッサの処理時間をできるだけ均等にすることで、１コマの描画に要する時間を短縮する。それにより、各プロセッサの利用効率を向上させる。具体的には、この第６の実施の形態では、直前の１コマの画像処理の際の、各プロセッサの処理時間や演算量を表す情報に基づいて、次の１コマの画像処理における、各プロセッサに割り当てる領域の大きさを調整する。そして、領域の大きさの調整によって、各プロセッサの処理時間をできるだけ均等にする。本発明の第６の実施の形態は、連続するコマ同士の類似性に着目したものである。

【０１１８】「各プロセッサの処理時間や演算量を表す情報（以下、「演算量情報」と呼ぶ）」としては、たとえば各プロセッサの消費処理時間、実行サイクル数、ローカルメモリのミス率等が挙げられる。１コマの描画においてこの演算情報量の値が他より大きいプロセッサは、連続するコマどうしの類似性により、その次のコマの描画においても大きな値を持つことが予想される。そして、この値が大きいことは、そのプロセッサの処理時

間が他のプロセッサと比べて長いことを意味する。そこで、１コマの描画処理においてこの値が大きいプロセッサに対しては描画処理における分担領域を小さくすることで、次の１コマにおける処理時間の短縮化を図る。ここでは、演算量情報として、各プロセッサの実行サイクル数を用いた場合について説明する。もちろん、消費処理時間、ローカルメモリのミス率等を用いても、何ら変わることはない。また、各プロセッサは、少なくとも１つの画素ブロック（画素の集合）の処理を担当する。各画素ブロックは、２×２（＝４）画素で構成されている。

【０１１９】１コマの描画処理における、各プロセッサが担当する画素ブロックの数は、直前描画情報に基づいて決定される。「直前描画情報」とは、直前の１コマの＊

0000 0000 0000 0000 0000 0000 0000 0000 0011 1011 1001 1010 1100 1010 0000 0000

“１”が立つ最上位ビットは、ＬＳＢから３０ビット目である。したがって、この場合、ビット縮退した値、つまり演算量情報は、３０となる。ここで、実際の実行サイクル数を６４ビットで表した場合、ビット縮退した値の最大値は６４となる。６ビットのパターンは２６＝６４通りある。したがって、それらのビットパターンによって、０から２６－１までの数値を十分表すことができる。

【０１２２】また、各プロセッサは複数の画素ブロックの処理を行うので、各プロセッサの演算量情報は各画素※

0000 0000 0000 0000 0000 0000 0000 0000 0010 0000 0000 0000 0000 0000 0000 0000

次の画素の演算量情報が２９の場合、ＬＳＢから２９ビット目に１を加算する。　★

0000 0000 0000 0000 0000 0000 0000 0000 0011 0000 0000 0000 0000 0000 0000 0000

次の画素の演算量情報が２９の場合、ＬＳＢから２９ビット目に１を加算する。　☆

0000 0000 0000 0000 0000 0000 0000 0000 0100 0000 0000 0000 0000 0000 0000 0000

図３５に、４つのプロセッサ（プロセッサ０、プロセッサ１、プロセッサ２およびプロセッサ３）で１６個の画素ブロックの描画を実行した場合における、各画素の描画に要した実行サイクル数およびそのビット縮退した値の例を示す。この描画処理では、各プロセッサそれぞれに、４つの画素ブロックが割り当てられている。図３５では、プロセッサ０の実行サイクル数は１６５００、プロセッサ１の実行サイクル数は７００００、プロセッサ２の実行サイクル数は５５００、プロセッサ３の実行サイクル数は７５００、である。これらの値から、図３５の場合の直前描画情報は、次のようになる。

【０１２７】（プロセッサ０，プロセッサ１，プロセッサ２，プロセッサ３）＝（２５０８８，１１４６８８，６６５６，９７２８）

＊描画処理における、各プロセッサの演算量情報を表す値の集合である。たとえばプロセッサが４つであれば直前描画情報は４つの整数の組となる。演算量情報として、実行サイクル数を採用した場合、その表示は実際の実行サイクル数をビット縮退したもので行う。データ送信量の低減のためである。具体的には、実行サイクル数を６４ビットの２進数の固定小数点数で表現した場合に、“１”が立つ最上位ビットの位置を演算量情報とする。小数点位置は、最下位ビット（least significant bit；ＬＳＢ）の右に固定される。

【０１２０】たとえば実行サイクル数が１０億である場合、６４ビットの２進数の固定小数点数は、次の通りである。

【０１２１】

0000 0000 0000 0000 0000 0000 0000 0000 0011 1011 1001 1010 1100 1010 0000 0000

※ブロックの処理に要した演算量情報の累積となる。演算量情報は、実行サイクル数をビット縮退したものである。したがって、演算量情報の累積は、１画素の演算量情報に、次の画素の演算量情報が示すビットに、“１”を加算することで行われる。

【０１２３】たとえば１画素の演算量情報が３０の場合、６４ビットの２進数の固定小数点数は、次の通りである。

【０１２４】

★【０１２５】

☆【０１２６】

この直前描画情報は、プロセッサ１の実行サイクル数が他と比べて突出して大きいことを示している。つまり、プロセッサ１の描画処理に要した時間が最も長かったことを意味している。そこで、次のコマの描画処理の際にはプロセッサ１に割り当てる画素ブロックの数を減じ、プロセッサ間の実行サイクル数の差を小さくする。具体的には、以下の式から各プロセッサが次のコマで担当する画素ブロックの数を決定する。

【０１２８】ｙ＝（１６／ｘ）／（（１／Ａ）＋（１／Ｂ）＋（１／Ｃ）＋（１／Ｄ））

ここで、ｙ：次のコマの描画処理の際に、各プロセッサに割り当てられる画素ブロックの数、ｘ：各プロセッサの演算量情報の値の和、Ａ：プロセッサ０の演算量情報の値、Ｂ：プロセッサ１の演算量情報の値、Ｃ：プロセ

ッサ２の演算量情報の値、Ｄ：プロセッサ３の演算量情報の値、である。

【０１２９】すなわち、次のコマで、各プロセッサに割り当てられる画素ブロックの数は、各プロセッサの直前のコマの演算量情報の値に反比例して配分される。図３５の場合、各プロセッサが次のコマの描画の際に割り当てられる画素ブロックの数は、次のようになる。

【０１３０】（プロセッサ０，プロセッサ１，プロセッサ２，プロセッサ３）＝（２．１１，０．４６，７．９７，５．４５）

ここで、画素ブロックの数は自然数でなければならない。そこで、次の調整方法が用いられる。

【０１３１】（１）小数点以下を四捨五入する。

【０１３２】（２）１未満の数値は１とする。

【０１３３】（３）４つの数値の合計を１６とする。

【０１３４】したがって、図３５の場合、各プロセッサが次のコマの描画の際に割り当てられる画素ブロックの数は、結局次のようになる。

【０１３５】（プロセッサ０，プロセッサ１，プロセッサ２，プロセッサ３）＝（２，１，８，５）

たとえば次のコマが直前のコマと同一の画像を描画するものである場合、この結果に基づいて、図３６に示すように、各プロセッサに割り当てられる画素ブロックの数を調整すれば良い。この場合、プロセッサ０の実行サイクル数は３５００、プロセッサ１の実行サイクル数は４００００、プロセッサ２の実行サイクル数は３８５００、プロセッサ３の実行サイクル数は１７５００、である。直前のコマの描画の際には最大実行サイクル数はプロセッサ１の７００００である。今回のコマの描画では最大実行サイクル数はプロセッサ１の４００００となり、３００００サイクルが削減される。

【０１３６】本発明の第６の実施の形態によれば、各プロセッサの描画時間をできるだけ均等にし、画像処理の効率化を図ることができる。

【０１３７】（第７の実施の形態）次に、本発明の第７の実施の形態について説明する。各プロセッサの計算時間は、担当領域の大きさおよびその中に存在するオブジェクトの数に比例する。したがって、担当する領域にオブジェクトが多いプロセッサと担当する領域にオブジェクトが少ないプロセッサとでは、その処理時間に差が生じる。そのため、１画像の描画処理時間は、最も処理の遅いプロセッサの処理時間に依存することになる。

【０１３８】本発明の第７の実施の形態では、担当領域の処理を先に終了したプロセッサが、他のプロセッサが担当する未処理領域を、さらに処理することで、画像処理の効率化を図るものである。以下、本発明の第７の実施の形態について図３７（Ａ）乃至（Ｄ）、図３８（Ａ）乃至（Ｃ）を用いて説明する。図３７（Ａ）乃至（Ｄ）は、４つのプロセッサ１，２，３，４によって画像処理される画面の例を示す図、図３８（Ａ）乃至（Ｃ）は、プロセッサの処理方向の例を示す図である。

【０１３９】図３７（Ａ）は、処理対象となる画面を示している。点線で区切られた領域（以下、「単位領域」と呼ぶ）は、各プロセッサが一度に取り扱うことのできる領域である。単位領域Ｌの処理に要する時間を１ｔ、単位領域Ｍの処理に要する時間を４ｔ、単位領域Ｎの処理に要する時間を８ｔとする。図３７（Ａ）に示すように、各プロセッサそれぞれには、６つの単位領域からなる領域が、割り当てられている。

【０１４０】各プロセッサの単位領域を処理する順番を決定する際に、次の２つの判断基準を用いる。

【０１４１】（１）プロセッサは基本的には担当領域の端に接しない単位領域に向かって処理を行なうこと。つまり、担当領域の内部から画面中心方向の端に接する単位領域に向かって処理を行うこと。

【０１４２】（２）プロセッサはできるだけ隣接する単位領域を順に処理していくこと。

【０１４３】ここで、その処理順番としては、たとえば図３８（Ａ）の渦巻状に進む順、図３８（Ｂ）の放射状に進む順、図３８（Ｃ）の螺旋状に進む順、あるいはこれらの組合わせにより進む順がある。

【０１４４】図３７（Ａ）では、各プロセッサが担当する単位領域はすべて、それぞれの担当領域の端に接している。そこで、たとえば図３７（Ｂ）に示す順で、各プロセッサの処理が進められる。

【０１４５】図３７（Ｃ）は、処理開始から時間９ｔが経過した時点の処理の進捗状況を示す図である。この時点では、プロセッサ２は担当領域の処理を終了している。プロセッサ２は最後に処理を行った単位領域に隣接する、他のプロセッサに割り当てられた領域の処理をさらに進める。ここではプロセッサ２が最後に処理を行った単位領域に隣接する領域はプロセッサ１に割り当てられた領域である。そして、その領域はこの時点では未処理である。したがって、プロセッサ２はプロセッサ１の割り当てられた未処理領域の処理を進める。

【０１４６】図３７（Ｄ）は、処理開始から時間１３ｔが経過した時点の処理の状況を示す図である。この時点ですべてのプロセッサの処理を終了し、画面全体の描画処理が終了している。各プロセッサが担当領域のみを処理するとした場合、画面全体の描画処理は最も処理時間の長いプロセッサに律束される。最も長い処理時間を要するのはプロセッサ１であり、その処理時間は１６ｔとなる。つまり、本例の場合、時間３ｔだけ短縮されていることがわかる。

【０１４７】本発明の第７の実施の形態によれば、負荷の軽いプロセッサが負荷の重いプロセッサの処理を補助することにより、各プロセッサの処理時間の平均化が図られる。その結果、画像処理の高速化が実現される。

【０１４８】

【発明の効果】本発明によれば、画像処理する計算機の

処理能力を向上させることができる。このため、高度な画像処理を安価なシステムで実現することができる。

【図面の簡単な説明】

【図１】本発明に係る画像処理方法で使用されるレイトレーシングを説明する概念図である。

【図２】図１のスクリーン２４を階層分割した例を示す図である。

【図３】本発明に係る画像処理装置のハードウェア構成を示すブロック図である。

【図４】（Ａ）は、２次元のオクツリーを説明する概念図、（Ｂ）は、３次元のオクツリーを説明する概念図である。

【図５】９つのオブジェクト（obj0，obj１，obj２，…，obj８）が存在する２次元空間に対して、２次元のオクツリーを適用した場合を説明する図である。

【図６】図５に示した階層構造の２次元空間をレイトレーシングに適用した場合を説明する図である。

【図７】本発明の第１の実施の形態に係る画像処理システムの構成を示す図である。

【図８】本発明の第１の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。

【図９】本発明の第２の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。

【図１０】図５に示した階層構造の２次元空間をビームトレーシングに適用した場合を説明する図である。

【図１１】（Ａ）は、本発明の第３の実施の形態に係る、論理的に分割された２つの領域を有するデータキャッシュを備えた、ローカルメモリの構造を示すブロック図、（Ｂ）は、本発明の第３の実施の形態に係る、物理的に分割された２つの領域を有するデータキャッシュを備えた、ローカルメモリの構造を示すブロック図である。

【図１２】（Ａ）は、一般的なフルアソシアティブ方式のキャッシュ構造を示す図、（Ｂ）は、一般的なダイレクトマップ方式のキャッシュ構造を示す図である。

【図１３】本発明の第３の実施の形態に係るオブジェクトデータキャッシュに格納されるオブジェクトデータ例を示す。

【図１４】本発明の第３の実施の形態に係るオブジェクトデータキャッシュの構造を示す図である。

【図１５】（Ａ）は、図１４に示したキャッシュメモリ７６の構成を示す図、（Ｂ）は、図１４に示したキャッシュテーブル７８の構成を示す図である。

【図１６】本発明の第３の実施の形態に係るオブジェクトデータキャッシュの、オブジェクトヒット時における動作を説明する図である。

【図１７】本発明の第３の実施の形態に係るオブジェクトデータキャッシュの、オブジェクトミス時における動作を説明する図である。

【図１８】本発明の第３の実施の形態に係るオブジェクトデータキャッシュに格納される、複数のオブジェクトデータ例を示す。

【図１９】（ａ）乃至（ｅ）は、本発明の第３の実施の形態に係るオブジェクトデータキャッシュが、図１８のデータを格納する際における、キャッシュメモリおよびキャッシュテーブルの内容の変化を示す図である。

【図２０】本発明の第４の実施の形態を説明する際に用いられる例である、共有メモリ型マルチプロセッサを示す図である。

【図２１】図２０のプロセッサ９２－０からのアクセス要求とプロセッサ９２－１からのアクセス要求の両方が、共有メモリ９０の「0x1000」番地への読み込みアクセスである場合を示す。

【図２２】（Ａ）は、図２０のプロセッサ９２－０からの１ワード（４バイト）アクセス要求が、共有メモリ９０の「0x1000」番地への読み込みアクセス、プロセッサ９２－１からの１ワードアクセス要求が、「0x1004」番地への読み込むアクセス、である場合であって、この２つのアクセス要求がコンバイン不可である場合を示す図、（Ｂ）は、図２０のプロセッサ９２－０からの１ワード（４バイト）アクセス要求が、共有メモリ９０の「0x1000」番地への読み込みアクセス、プロセッサ９２－１からの１ワードアクセス要求が、「0x1004」番地への読み込むアクセス、である場合であって、この２つのアクセス要求が拡張コンバイン可能である場合を示す図である。

【図２３】本発明の第４の実施の形態に係る待ち行列システムの構成を示す図である。

【図２４】図２３の調停部の構成を示す図である。

【図２５】図２３の比較部の構成を示す図である。

【図２６】図２３の待ち行列部の構成を示す図である。

【図２７】図２６のコンバイン情報テーブルを示す図であって、プロセッサ０からアドレス「0x1010」への４word-Burst-readと、プロセッサ１からアドレス「0x1008」への２word-Burst-readと、プロセッサ２およびプロセッサ３からアドレス「0x1000」への２word-Burst-readと、が拡張コンバインされている場合を示す図である。

【図２８】図２６のコンバイン情報テーブルを示す図であって、プロセッサ２およびプロセッサ５からアドレス「0x1000」へのByte-readと、プロセッサ４からアドレス「0x1002」へのByte-Readと、が拡張コンバインされている場合を示す図である。

【図２９】図２３の待ち行列部に備えられる、バッファメモリの構成を示す図である。

【図３０】本発明の第５の実施の形態に係る画像処理方法の処理手順を示すフローチャートである。

【図３１】本発明の第５の実施の形態に係る画像処理方法の処理手順を示すフローチャートであって、処理対象の３次元空間がオクツリーで空間分割されている場合を

示すフローチャートである。

【図３２】本発明の第５の実施の形態を説明する際に用いられる例である、２次元空間内のオブジェクトとの交差判定を行なう場合を示す図である。

【図３３】（ａ）乃至（ｄ）は、図３２の交差判定を行なう際の、ステータステーブルの内容の変化を示す図である（その１）。

【図３４】（ｅ）乃至（ｇ）は、図３２の交差判定を行なう際の、ステータステーブルの内容の変化を示す図である（その２）。

【図３５】本発明の第６の実施の形態を説明する際に用いられる例である、４つのプロセッサ（プロセッサ０、プロセッサ１、プロセッサ２およびプロセッサ３）で１６個の画素ブロックの描画を実行した場合における、各画素の描画に要した実行サイクル数およびそのビット縮退した値を示す図である。

【図３６】図３５の４つのプロセッサに割り当てられる画素ブロックの、調整後の数を示す図である。

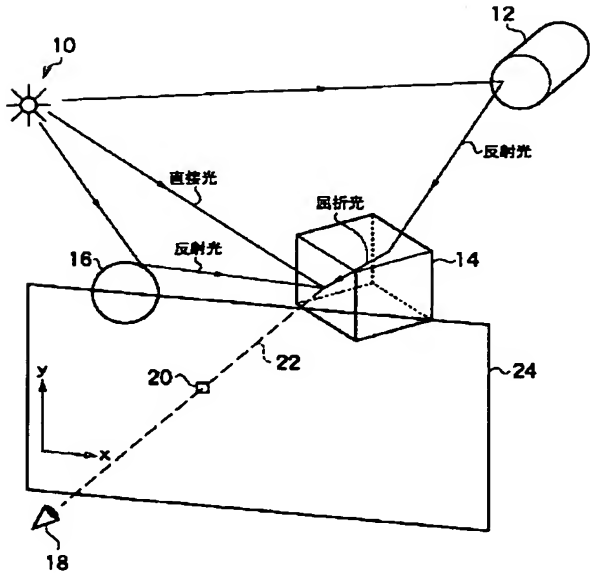
【図３７】（Ａ）乃至（Ｄ）は、本発明の第７の実施の形態を説明する際に用いられる例である、４つのプロセッサ１，２，３，４によって画像処理される画面の、時間と共に変化する処理状況を示す図である。

【図３８】（Ａ）は、図３７のプロセッサ１，２，３，４の処理方向が渦巻状である場合を示す図、（Ｂ）は、放射状である場合を示す図、（Ｃ）は、螺旋状である場合を示す図である。

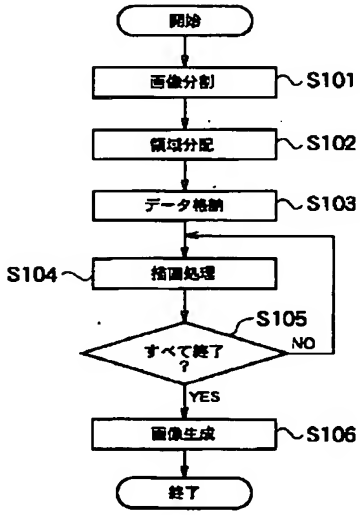
【符号の説明】

１０　光源
１２，１４，１６　オブジェクト
１８　視点
＊２０　画素
２２　光線（レイ）
２４　スクリーン
２６　サブ・スクリーン
２８　サブ・サブ・スクリーン
３０，８０　主メモリ
３２，９２　プロセッサ
３４，３６，３８，４８，６４，６８　プロセッサコア
４０，４２，４４　ローカルメモリ
４６，６２，９０　共有メモリ
５０，５４，６０，９４　ネットワーク
５２　チップ
５６　ボード
５８　コンピュータ
６６　データキャッシュ
７０　階層構造データキャッシュ
７２，７４　オブジェクトデータキャッシュ
７６　キャッシュメモリ
７８　キャッシュテーブル
８２　オブジェクトトランスファコントローラ
８４　ＣＰＵ
８６　加算器
８８　レジスタ
９６　調停部
９８　比較部
１００　待ち行列部
１０２　マルチキャスト部
１０４　マルチプレクサ
１０６　アクセステーブル
＊１０８　２次元空間

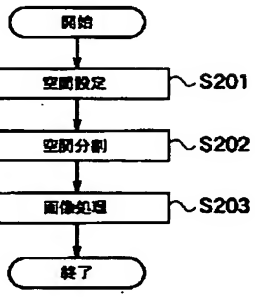
【図１】



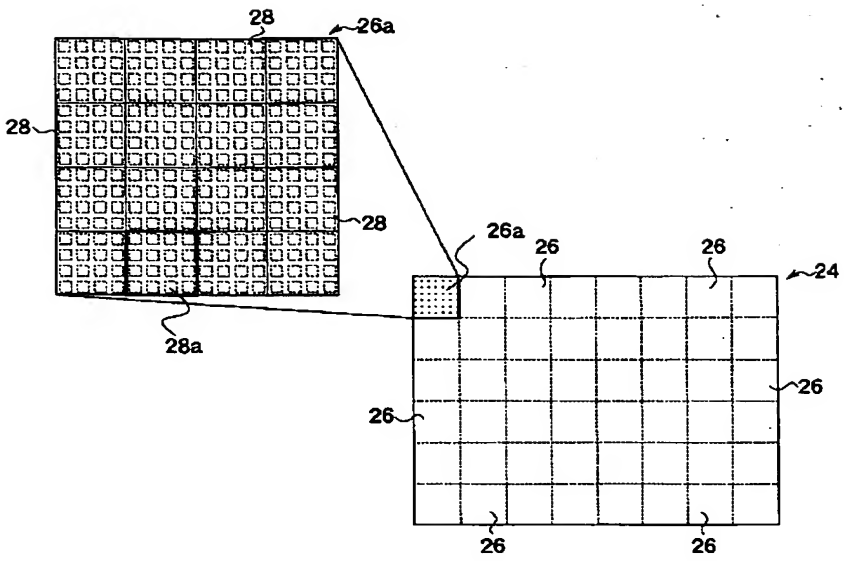
【図８】



【図９】



【図２】



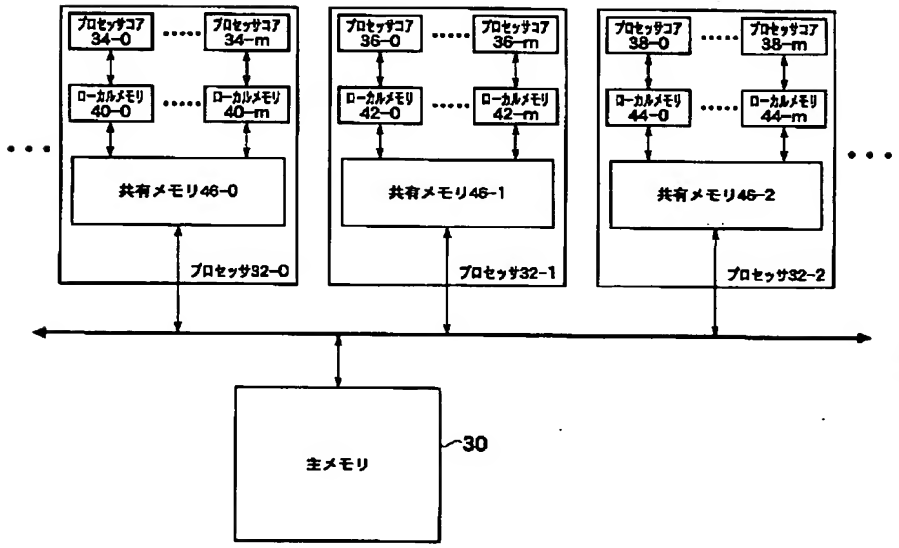
【図１３】

ADDRESS	DATA
0x00	sphere
0x01	x coordinate
0x02	y coordinate
0x03	z coordinate
0x04	radius
0x05	color
0x06	ambient
0x07	diffuse
0x08	specular
0x09	----------
0x0a	----------
0x0b	----------
0x0c	----------

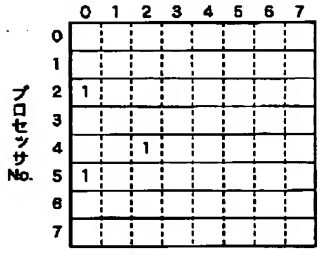
【図１８】

	start address	size
object1:	0x2000	0x04
object2:	0x2004	0x08
object3:	0x200c	0x06
object4:	0x2002	0x06
object5:	0x200a	0x08
object6:	0x200e	0x04
object7:	0x2012	0x04
object8:	0x2018	0x06

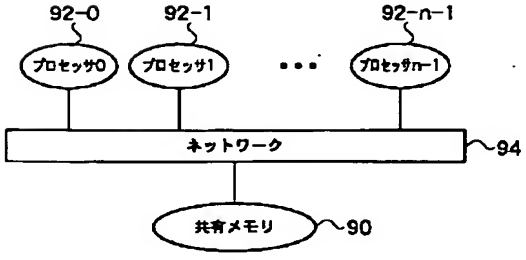
【図３】



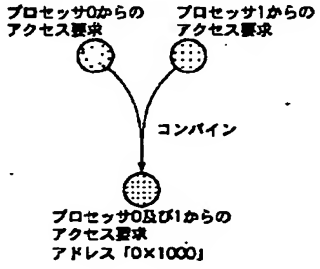
【図２８】



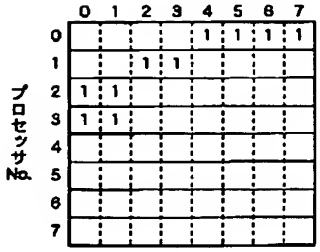
【図２０】



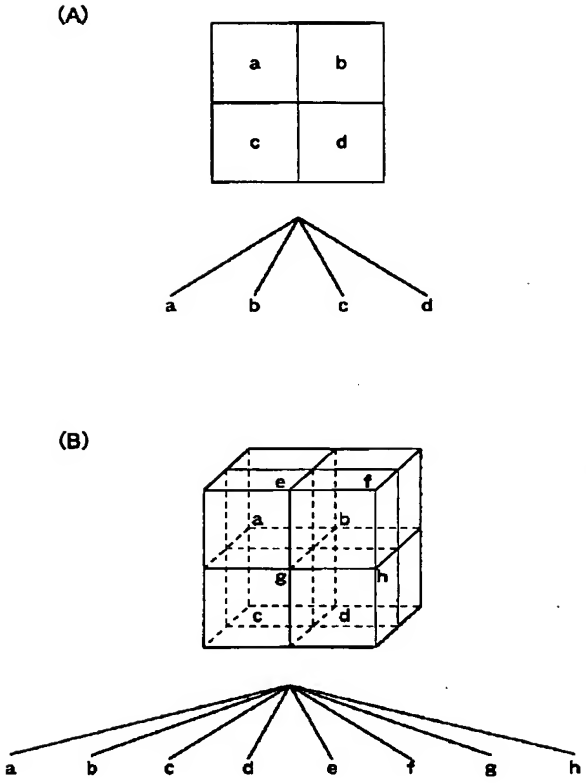
【図２１】



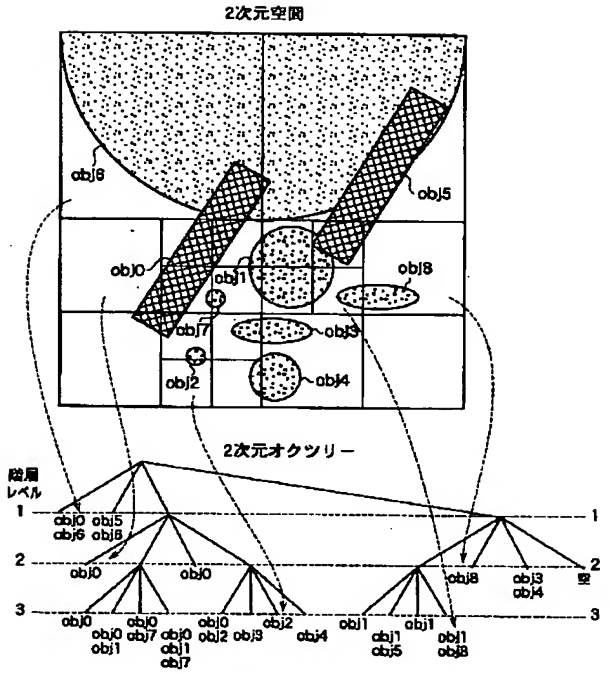
【図２７】



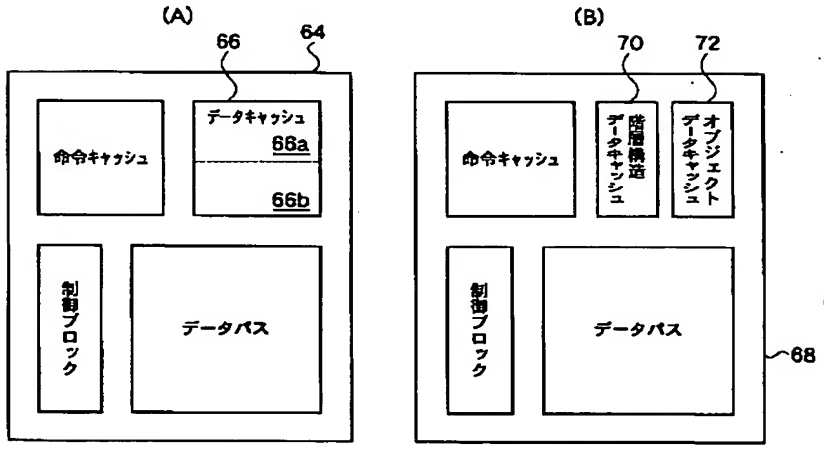
【図４】


(A)
a
b
c
d
a
b
c
d
(B)
e
f
a
b
g
h
c
d
a
b
c
d
e
f
g
h

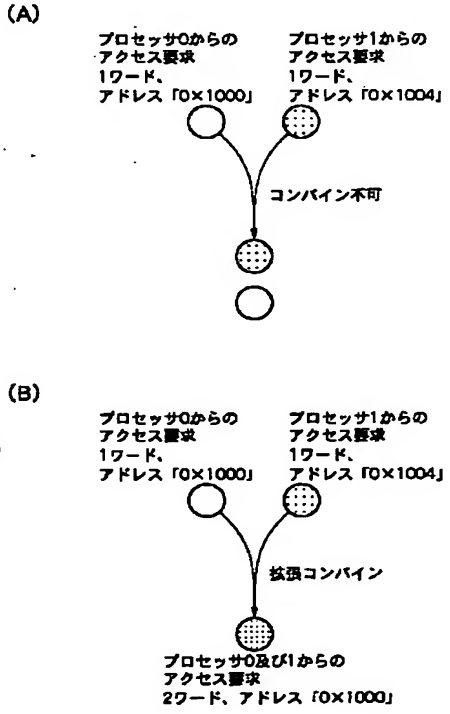
【図５】


2次元空間
obj6
obj5
obj0
obj1
obj8
obj7
obj3
obj2
obj4
2次元オクツリー
階層レベル
1
2
3
obj0 obj5 obj6 obj8
obj0
obj0
obj8
obj3 obj4
空
obj0
obj0 obj7
obj0 obj1 obj7
obj0 obj2
obj3
obj2
obj4
obj1
obj1
obj1 obj5
obj1 obj8

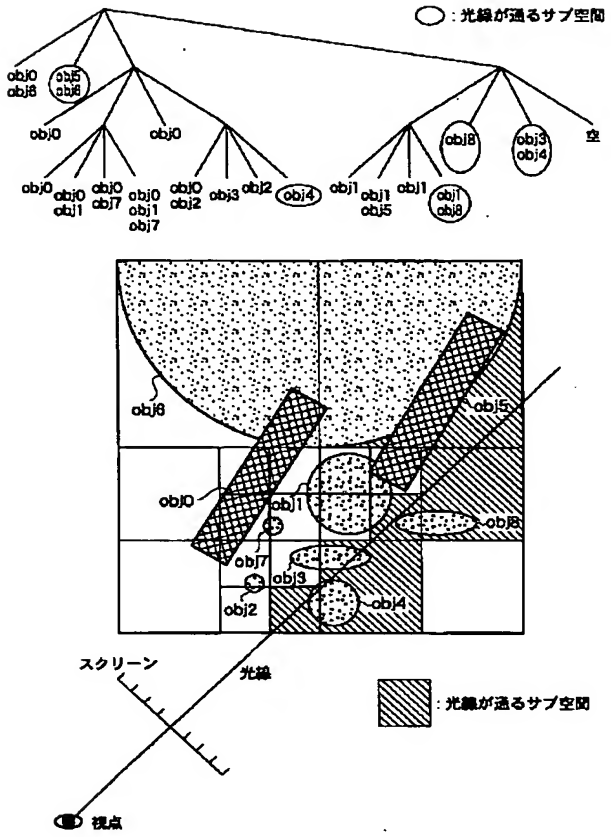
【図１１】


(A)
66
64
命令キャッシュ
データキャッシュ
66a
66b
制御ブロック
データパス
(B)
70
72
命令キャッシュ
階層構造データキャッシュ
オブジェクトデータキャッシュ
制御ブロック
データパス
68

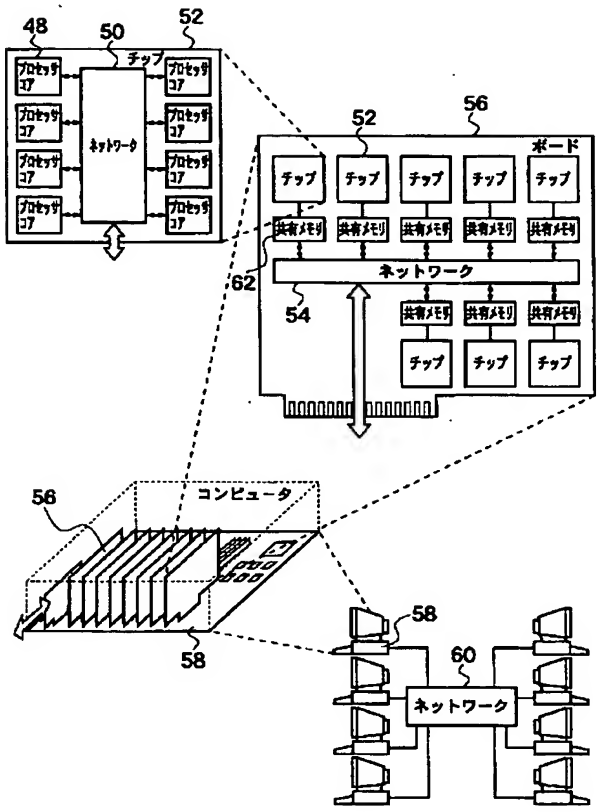
【図２２】


(A)
プロセッサ0からのアクセス要求 1ワード、アドレス「0×1000」
プロセッサ1からのアクセス要求 1ワード、アドレス「0×1004」
コンバイン不可
(B)
プロセッサ0からのアクセス要求 1ワード、アドレス「0×1000」
プロセッサ1からのアクセス要求 1ワード、アドレス「0×1004」
拡張コンバイン
プロセッサ0及び1からのアクセス要求 2ワード、アドレス「0×1000」

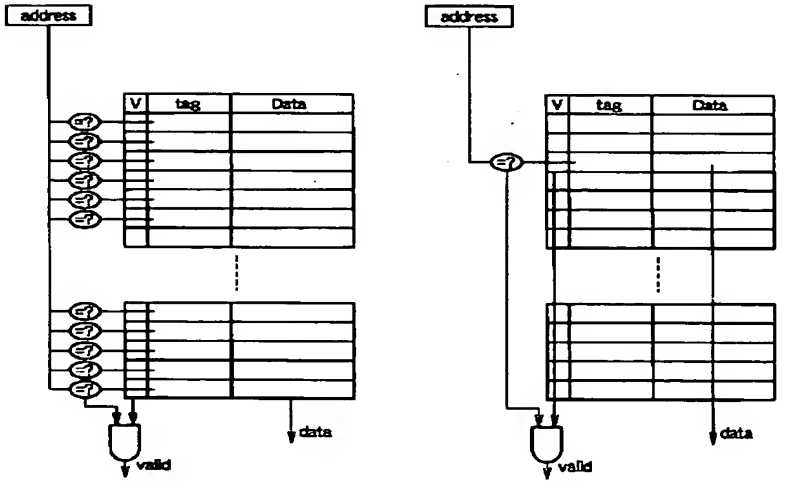
【図６】


：光線が通るサブ空間
obj0
obj8
obj5
obj6
obj0
obj0
空
obj8
obj3
obj4
obj0
obj1
obj0
obj7
obj0
obj1
obj7
obj0
obj2
obj3
obj2
obj4
obj1
obj5
obj1
obj1
obj1
obj8
obj8
obj5
obj0
obj1
obj8
obj7
obj3
obj2
obj4
スクリーン
光線
：光線が通るサブ空間
視点

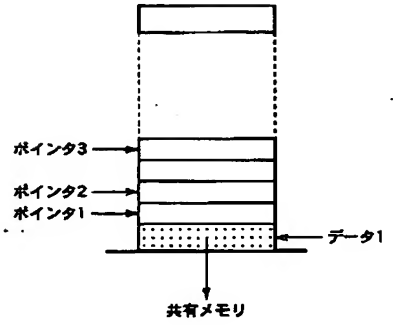
【図７】


48
50
52
チップ
プロセッサコア
ネットワーク
52
56
ボード
チップ
共有メモリ
ネットワーク
62
54
56
コンピュータ
58
58
60
ネットワーク

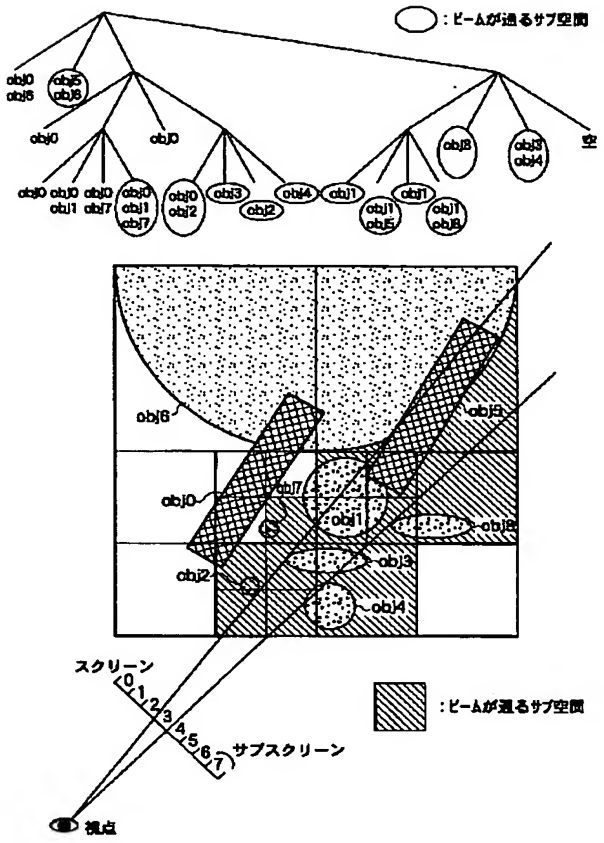
【図１２】


（A）
address
V
tag
Data
=?
data
valid
（B）
address
V
tag
Data
=?
data
valid

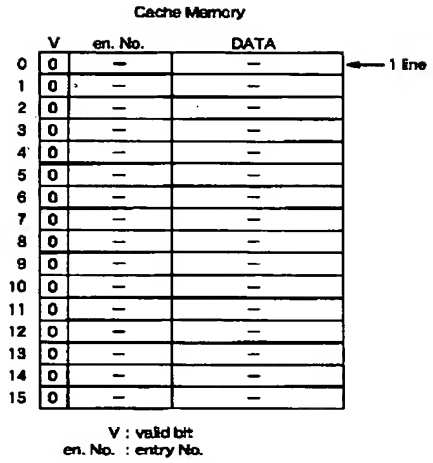
【図２９】

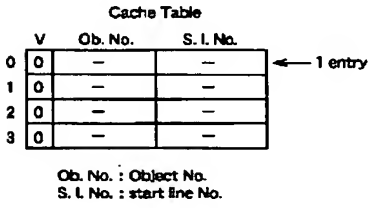

ポインタ3
ポインタ2
ポインタ1
データ1
共有メモリ

【図１０】

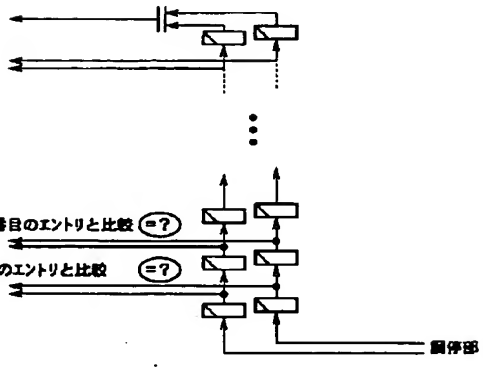


【図１５】

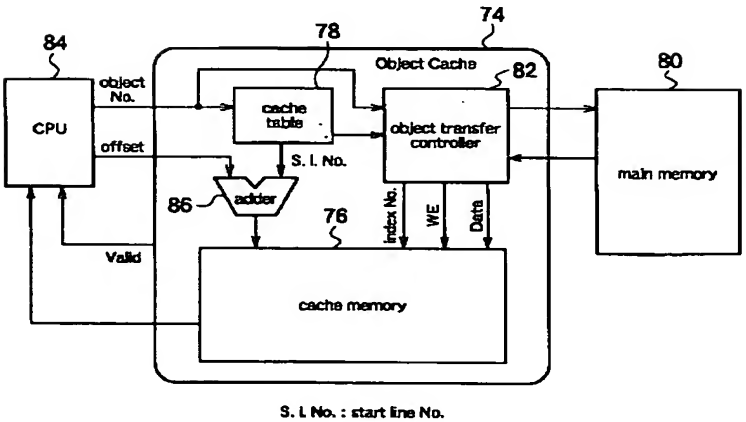




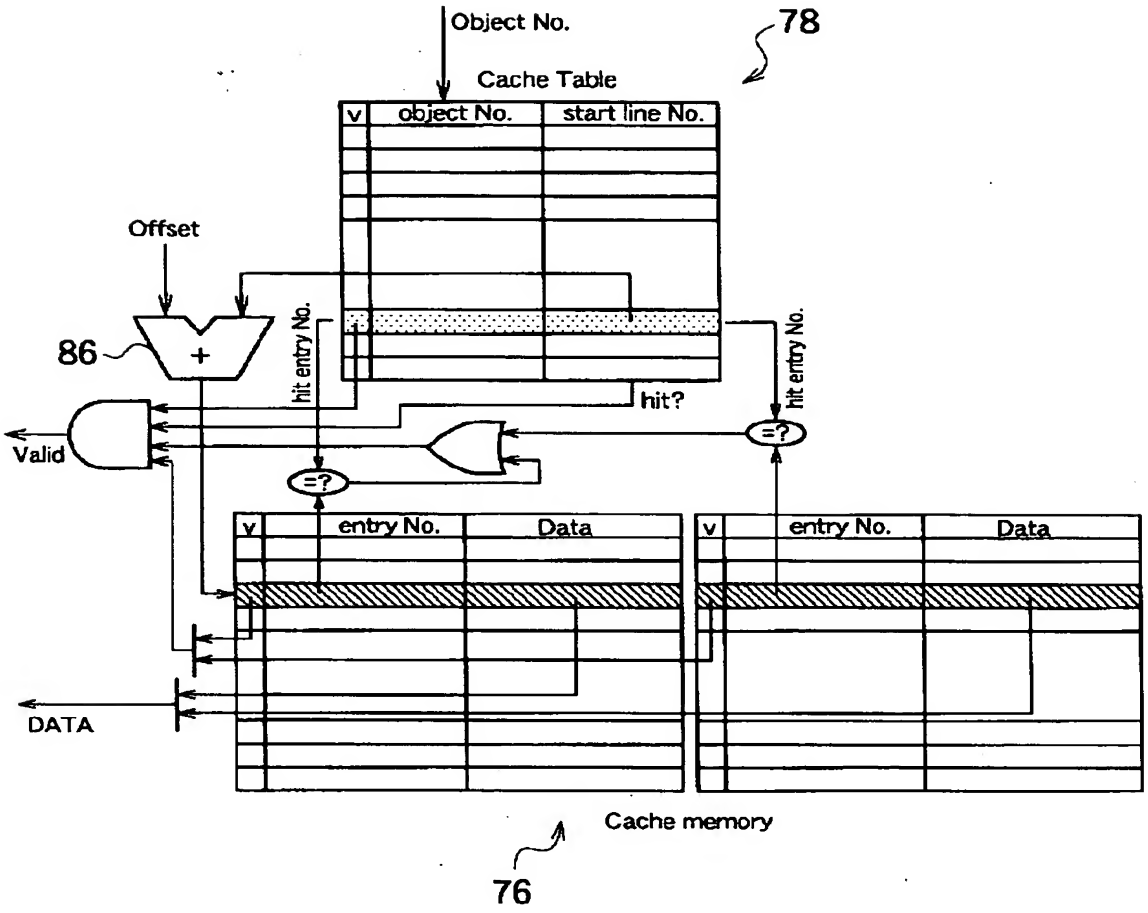
【図２５】



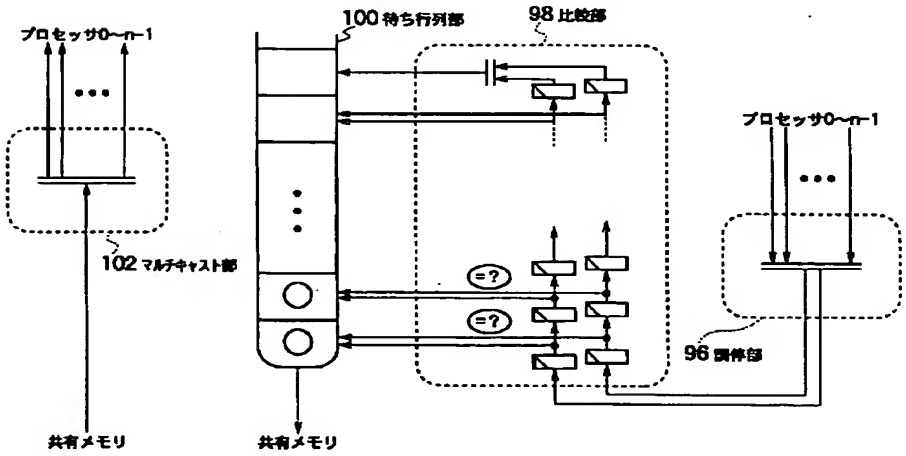
【図１４】



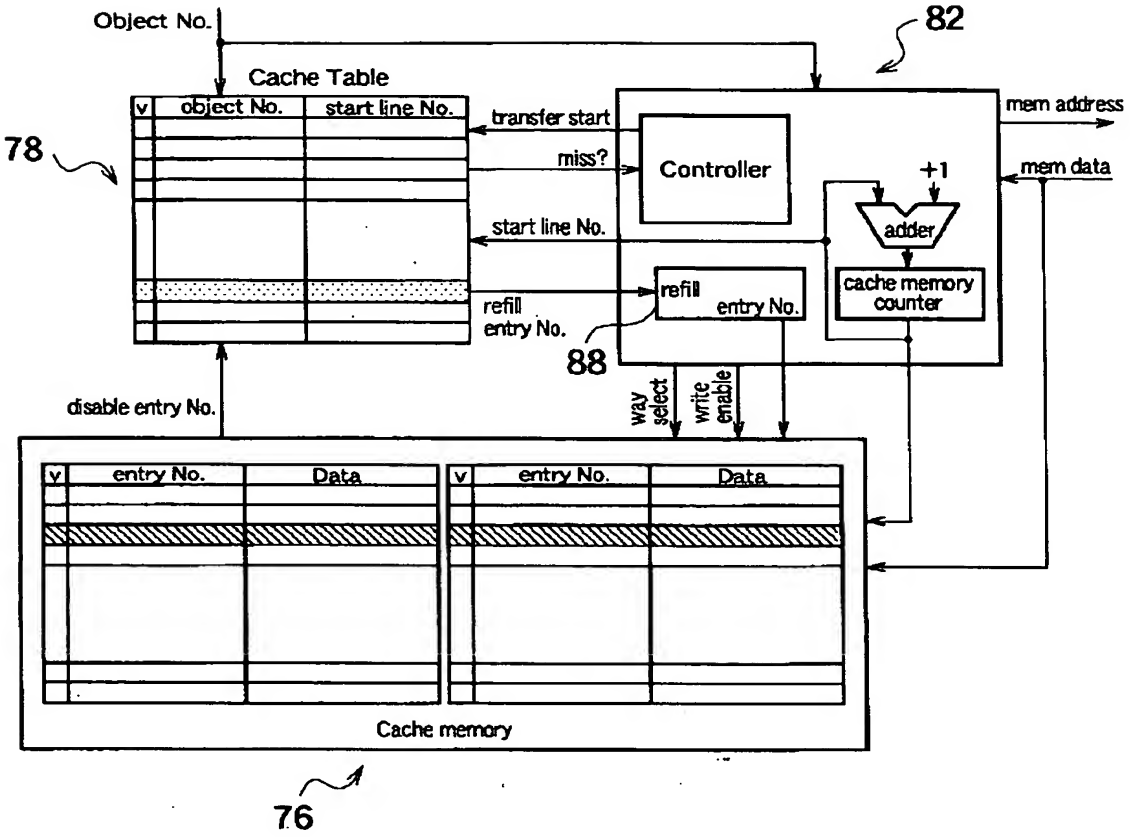
【図１６】


Object No.
78
Cache Table
v
object No.
start line No.
Offset
86
+
hit entry No.
hit?
hit entry No.
Valid
=?
=?
v
entry No.
Data
v
entry No.
Data
DATA
Cache memory
76

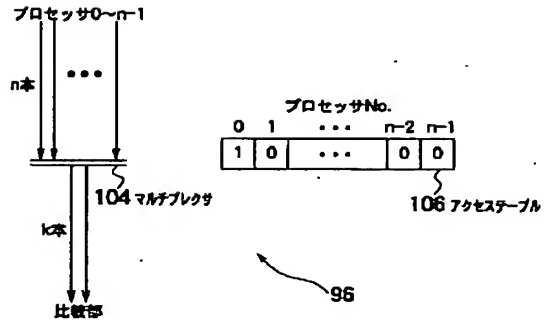
【図２３】


プロセッサ0～n-1
100 待ち行列部
98 比較部
プロセッサ0～n-1
102 マルチキャスト部
=?
=?
96 調停部
共有メモリ
共有メモリ

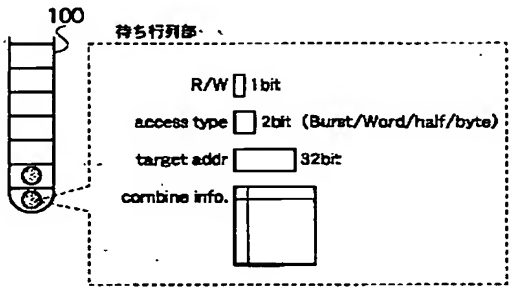
【図１７】


Object No.
82
Cache Table
v
object No.
start line No.
78
transfer start
miss?
start line No.
refill entry No.
Controller
+1
adder
cache memory counter
refill entry No.
88
mem address
mem data
disable entry No.
way select
write enable
v
entry No.
Data
v
entry No.
Data
Cache memory
76

【図２４】


プロセッサ0~n-1
n本
104 マルチプレクサ
k本
比較部
プロセッサNo.
0 1 ・・・ n-2 n-1
1 0 ・・・ 0 0
106 アクセステーブル
96

【図２６】


100
待ち行列部
R/W 1bit
access type 2bit (Burst/Word/half/byte)
target addr 32bit
combine info.

【図１９】

	v	ob. No.	S. I. No.
0	0	–	–
1	0	–	–
2	0	–	–
3	0	–	–

	v	en. No.	DATA
0	0	–	–
1	0	–	–
2	0	–	–
3	0	–	–
4	0	–	–
5	0	–	–
6	0	–	–
7	0	–	–
8	0	–	–
9	0	–	–
10	0	–	–
11	0	–	–
12	0	–	–
13	0	–	–
14	0	–	–
15	0	–	–

(a)

	v	ob. No.	S. I. No.
0	1	1	0
1	0	–	–
2	0	–	–
3	0	–	–

	v	en. No.	DATA
0	1	0	obj1-data1
1	1	0	obj1-data2
2	1	0	obj1-data3
3	1	0	obj1-data4
4	0	–	–
5	0	–	–
6	0	–	–
7	0	–	–
8	0	–	–
9	0	–	–
10	0	–	–
11	0	–	–
12	0	–	–
13	0	–	–
14	0	–	–
15	0	–	–

(b)

	v	ob. No.	S. I. No.
0	1	1	0
1	1	2	4
2	0	–	–
3	0	–	–

	v	en. No.	DATA
0	1	0	obj1-data1
1	1	0	obj1-data2
2	1	0	obj1-data3
3	1	0	obj1-data4
4	1	1	obj2-data1
5	1	1	obj2-data2
6	1	1	obj2-data3
7	1	1	obj2-data4
8	1	1	obj2-data5
9	1	1	obj2-data6
10	1	1	obj2-data7
11	1	1	obj2-data8
12	0	–	–
13	0	–	–
14	0	–	–
15	0	–	–

(c)

	v	ob. No.	S. I. No.
0	1	1	0
1	1	2	4
2	1	3	12
3	0	–	–

	v	en. No.	DATA
0	0	0	obj1-data1
1	0	0	obj1-data2
2	0	0	obj1-data3
3	0	0	obj1-data4
4	0	1	obj2-data1
5	0	1	obj2-data2
6	0	1	obj2-data3
7	0	1	obj2-data4
8	0	1	obj2-data5
9	0	1	obj2-data6
10	0	1	obj2-data7
11	0	1	obj2-data8
12	0	2	obj3-data1
13	0	2	obj3-data2
14	0	2	obj3-data3
15	0	2	obj3-data4

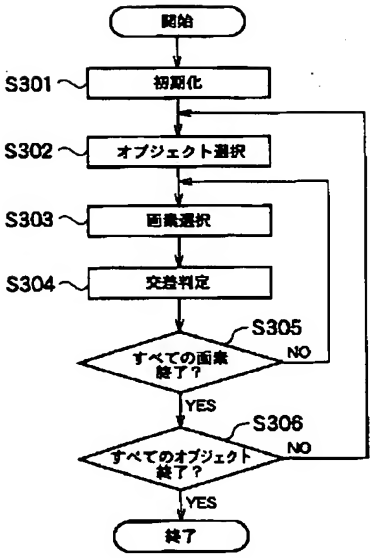
(d)

	v	ob. No.	S. I. No.
0	0	1	0
1	1	2	4
2	1	3	12
3	0	–	–

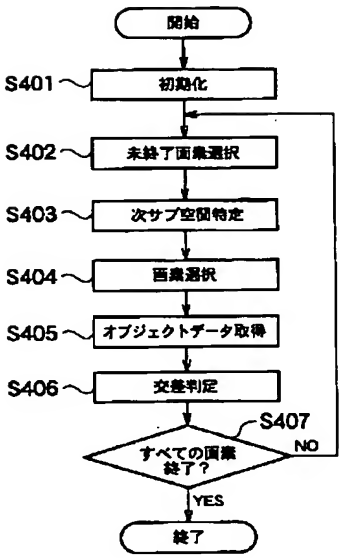
	v	en. No.	DATA
0	1	2	obj3-data5
1	1	2	obj3-data6
2	1	0	obj1-data3
3	1	0	obj1-data4
4	1	1	obj2-data1
5	1	1	obj2-data2
6	1	1	obj2-data3
7	1	1	obj2-data4
8	1	1	obj2-data5
9	1	1	obj2-data6
10	1	1	obj2-data7
11	1	1	obj2-data8
12	1	2	obj3-data1
13	1	2	obj3-data2
14	1	2	obj3-data3
15	1	2	obj3-data4

(e)

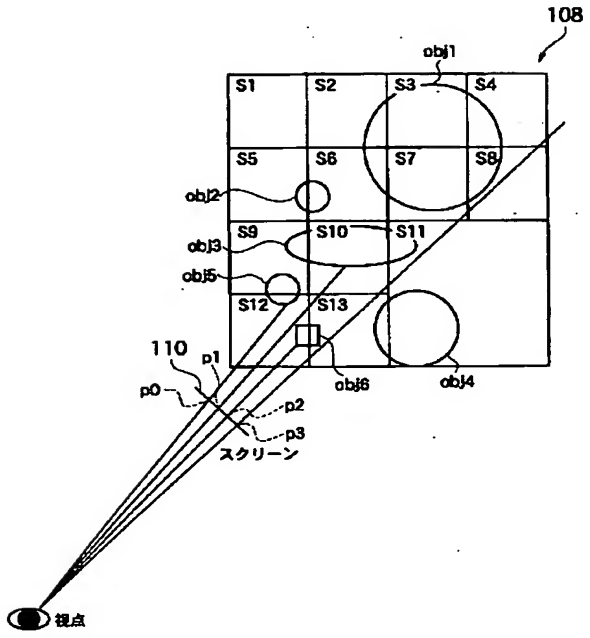
【図３０】



【図３１】



【図３２】



【図３３】

(a)

画素No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	No	—	—	—	∞
p1	No	—	—	—	∞
p2	No	—	—	—	∞
p3	No	—	—	—	∞

(b)

画素No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	No	S12	—	—	∞
p1	No	S12	—	—	∞
p2	No	S12	—	—	∞
p3	No	S12	—	—	∞

(c)

画素No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	No	S12	—	—	∞
p2	Yes	S12	obj8	(x2,y2)	d2
p3	No	S12	—	—	∞

(d)

画素No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	No	S13	—	—	∞
p2	Yes	S12	obj8	(x2,y2)	d2
p3	No	S13	—	—	∞

【図３４】

(e)

画素No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	No	S10	—	—	∞
p2	Yes	S12	obj8	(x2,y2)	d2
p3	No	S10	—	—	∞

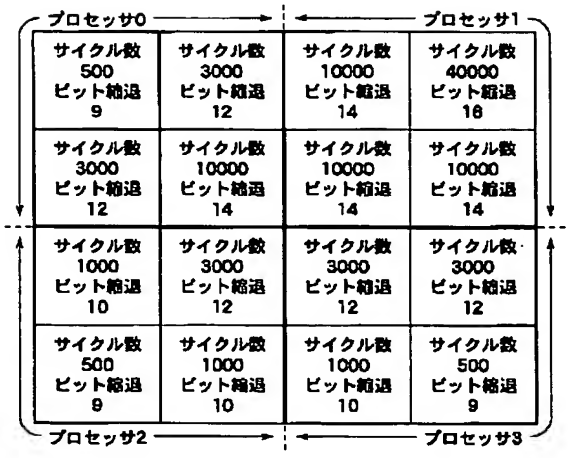
(f)

画素No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	Yes	S10	obj10	(x3,y3)	d3
p2	Yes	S12	obj8	(x2,y2)	d2
p3	No	S10	—	—	∞

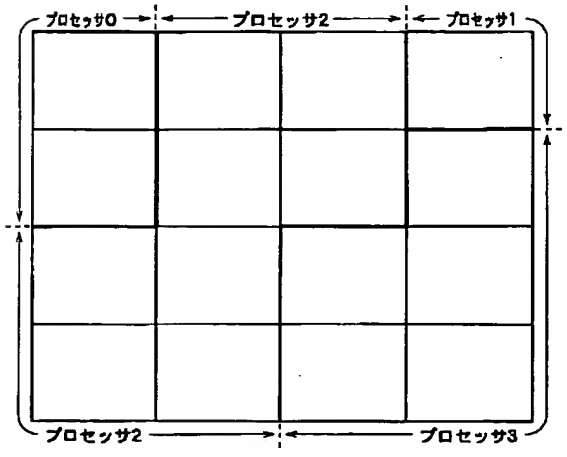
(g)

画素No.	終了	サブ空間	オブジェクト	交点座標	距離
p0	Yes	S12	obj5	(x1,y1)	d1
p1	Yes	S10	obj10	(x3,y3)	d3
p2	Yes	S12	obj8	(x2,y2)	d2
p3	Yes	—	—	—	∞

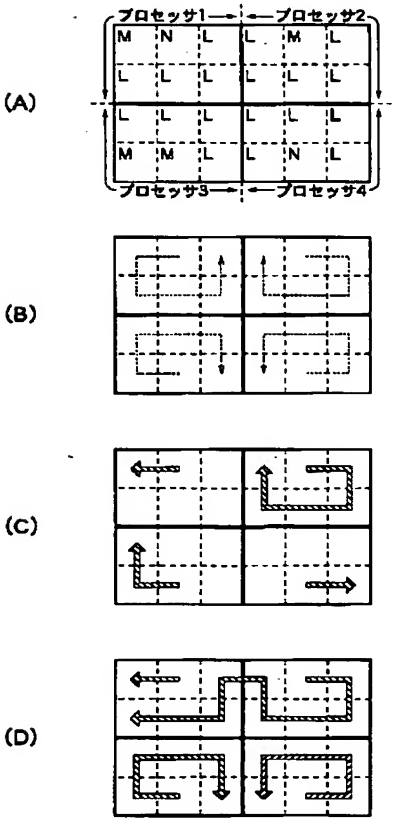
【図３５】



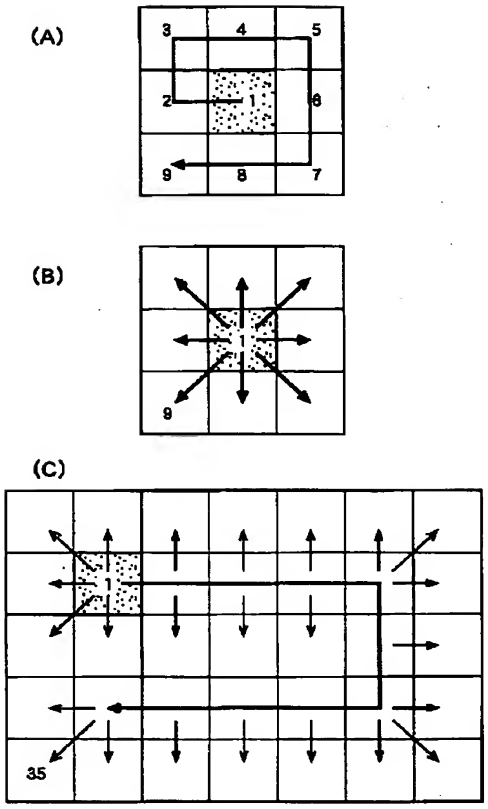
【図３６】



【図３７】



【図３８】



フロントページの続き

(72)発明者　安川　英樹
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝マイクロエレクトロニクスセンター内
(72)発明者　渡辺　幸男
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝マイクロエレクトロニクスセンター内
(72)発明者　亀井　貴之
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝マイクロエレクトロニクスセンター内
(72)発明者　雨坪　孝尚
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝マイクロエレクトロニクスセンター内